SAND2001-3512J
Unlimited Release
Printed March 2002

Iterative Solvers and Preconditioners for Fully-coupled Finite Element Formulations of Incompressible Fluid Mechanics and Related Transport Problems

P. R. Schunk, R. R. Rao, A. C. Sun, T. A. Baer
Multiphase Transport Processes

M. A. Heroux
Computational Mathematics and Algorithms

S. R. Subia
Thermal/Fluid Computational Engineering Sciencess

Sandia National Laboratories
P. O. Box 5800
Albuquerque, New Mexico 87185–0834

Abstract

Finite element discretization of fully-coupled, incompressible flow problems with the classic mixed velocity-pressure interpolation produces matrix systems that render the best and most robust iterative solvers and preconditioners ineffective. The indefinite nature of the discretized continuity equation is the root cause and is one reason for the advancement of pressure penalty formulations, least-squares pressure stabilization techniques, and pressure projection methods. These alternatives have served as admirable expedients and have enabled routine use of iterative matrix solution techniques; but all remain plagued by exceedingly slow convergence in the corresponding nonlinear problem, lack of robustness, or limited range of accuracy. The purpose of this paper is to revisit matrix systems produced by this old mixed velocity-pressure formulation with two approaches: (1) deploying well-established tools consisting of matrix system reordering, GMRES, and ILU preconditioning on modern architectures with substantial distributed or shared memory, and (2) tuning the preconditioner by managing the condition number using knowledge of the physical causes leading to the large condition number. Results obtained thus far using these simple techniques are very encouraging when measured against the reliability (not efficiency) of a direct matrix solver. Here we demonstrate routine solution for an incompressible flow problem using the Galerkin finite element method, Newton-Raphson iteration, and the robust and accurate LBB element. We also critique via an historical survey the limitations of pressure-stabilization strategies and all other commonly used alternatives to the mixed formulation for acceleration of iterative solver convergence. The performance of the new iterative solver approaches on other classes of problems, including fluid-structural interaction, multi-mode viscoelasticity, and free surface flow is also demonstrated.

1 Acknowledgements

The authors would like to acknowledge the support of the LDRD program which funded development of the preconditioners for GOMA .

Contents

List of Figures

List of Tables

Intentionally Left Blank

2 Introduction

The quest for robust and efficient strategies to solve large, sparse matrix systems has been an active area of research for decades. Even today numerous advances in matrix solver technology constantly appear in the literature, most being based on preconditioned iterative methods. Relatively few advances address direct methods, viz. methods based on Gaussian elimination, because of precipitous cost growth for large-bandwidth problems. Moreover, little attention has been given to the application of iterative solvers to matrices that are indefinite, i.e. those that possess unbounded large or negative eigenvalues. The future of implicit numerical techniques hinges on making iterative matrix solvers generally applicable since they are the only economical means, and perhaps the only feasible means, of solving large problems.

This paper addresses the current state and recent progress towards efficient, robust iterative matrix solution techniques for indefinite, nondiagonally dominant matrix systems. Matrices possessing these challenging traits arise in the mixed-interpolation implicit finite element discretization of the Navier-Stokes system of equations, the focus application in this paper. Peculiar to this equation set is the well-known continuity equation, which enforces incompressibility of the fluid. This equation, discretized by whatever means, has challenged numerical methods researchers for years and has dictated the course of development of efficient solution strategies. In all cases alternatives to the mixed-interpolation, fully-coupled approach have been developed with some compromise. Indeed our effort to push into production alternative formulations that can readily exploit the power of iterative methods has suffered from slower convergence rates, lower accuracy and, most importantly, a lack of consistency in convergence behavior from problem to problem. The last compromise is particularly noteworthy as it often presents the greatest barrier to progress of computer-aided engineering analysis.

The classical mixed velocity-pressure interpolation Galerkin finite element method (GFEM), often referred to as "mixed v-p form," was the workhorse of incompressible flow solvers in the 1970s and 1980s and has proved highly successful for two-dimensional and small three-dimensional problems, offering high order of accuracy and strong convergence rates (Reddy and Gartling [47]). The hallmark of this approach is the discretization of the continuity equation in a manner adherent to certain mathematical constraints (see Section 3), and the ability to simultaneously solve the equation with the discretized momentum equations in either a Picard or Newton-Raphson iterative scheme. The discretized continuity equation is the known cause of "indefiniteness" in the resulting Jacobian matrix, leading to unbounded large and negative eigenvalues. Despite this poor matrix conditioning, most two-dimensional problems with small bandwidth are amenable to efficient solution with optimized variations of classical Gaussian elimination. For example, the frontal solver (Hood [34]) and skyline method (Hasbani and Engelman [32]) allow for memory-efficient solution and in fact are still used quite often today. But even tremendous technological advances in computer processor speed and random access memory capacity will never make direct methods a viable alternative for large three-dimensional problems. This is one reason why many researchers have abandoned the traditional

mixed-interpolation approach in favor of alternatives that allow the ready use of iterative solvers. These alternative formulations, including penalty methods (e.g. Zienkiewicz [64]), pressure stabilization methods (Hughes et al. [37]) and pressure projection methods (Haroutunian et al. [31]), bring "definiteness" to the matrix system.

This paper revisits the fully-coupled, mixed-interpolation approach with the goal of effectively applying iterative solver techniques to the resulting matrix system. Preconditioning the matrix system is the primary tactic pursued in this work. Towards this goal we will take two approaches: (1) deploying well-established tools consisting of matrix system reordering, GMRES [52], and ILU preconditioning on modern architectures with substantial distributed or shared memory, and (2) tuning the preconditioner with diagonal perturbations using knowledge of the physical causes leading to the large condition number.

This paper is organized as follows: the mathematical description of three prototypical problems is given in Section 3.1. All three problems involve incompressible flow but are distinguished by additional complexities, e.g. capillary free-surfaces, fluid-structural interactions, complex and time-dependent material behavior, and multiple material/heterogeneous physics. This section also presents the basic essentials of the Galerkin/finite element method (GFEM) of discretization and discusses how coupling this approach with Newton-Raphson nonlinear iteration affects the conditioning of the matrix system that results. An historical review of the various alternate formulations taken to solve these systems will be covered in Section 4. Section 5 reviews the generalized minimum residual (GMRES) iterative solver technique and the concept of preconditioning, focusing mainly on our new preconditioning strategy. Application and performance of the iterative solvers and preconditioners on three test problems will be presented in Section 6. The first two test problems consist of basic Navier-Stokes equations in the simple three-dimensional lid-driven cavity and four-to-one contraction configurations. The next test problems involve fluid/structural interactions, a phenomenon that has been found to lead to severe matrix ill-conditioning. We then add in free and moving boundaries, and the constraint boundary conditions that distinguish the boundaries, as a complicating feature. Finally we will consider a rheologically complex flow, which even in two dimensions is characterized by a very large bandwidth that is large enough to render direct solvers too memory-intensive to be practical.

3 The Incompressible Flow Problem with Additional Complications

3.1 Governing Differential Equations

Although the applications we address in this paper focus on low- to moderate-speed flow, our mathematical description will be broadened to include other complicating features, including fluid-structure interaction and viscoelastic fluid mechanics. We

begin with the continuity equation

$$\nabla \cdot \mathbf{v} = 0 \tag{1}$$

and the fluid momentum equation

$$\rho \frac{\partial \mathbf{v}}{\partial t} = -\rho(\mathbf{v} - \mathbf{v}_s) \cdot \nabla \mathbf{v} + \mathbf{f} + \nabla \cdot \mathbf{T} \quad , \tag{2}$$

which constitute the equations for incompressible laminar flow. Note, if the stress is chosen to be Newtonian, these equations reduce to the well-known Navier-Stokes equations. Here $\mathbf{v}$ is the mass averaged fluid velocity, $\mathbf{v}_s$ is the mesh velocity, t is the time variable, ρ is the liquid density, $\mathbf{f}$ is the body force per unit volume, and $\mathbf{T}$ is the stress tensor. The mesh velocity, $\mathbf{v}_s$, is generally unrelated to the velocity of the fluid $\mathbf{v}$. Its effect is confined to the advection term (first term on the left hand side of Eq. 2). This term arises in so-called arbitrary Lagrangian Eulerian (ALE) formulations and is employed in this work for free and moving boundary problems with moving meshes. Details concerning the origins of ALE can be obtained from several sources (Huerta and Liu [36]). Equations governing the mesh position and/or velocity as required by ALE formulations, are discussed below. Note that we have allowed for transient flow, although some of our examples employ the steady-flow equations, e.g. Eq. 2 without the temporal derivatives.

The Cauchy stress tensor $\mathbf{T}$ for a Newtonian liquid is given by

$$\mathbf{T} = -p\mathbf{I} + 2\mu\mathbf{D}, \tag{3}$$

where p is the fluid pressure, $\mathbf{I}$ is the identity tensor, μ is the fluid viscosity and

$$\mathbf{D} = \frac{1}{2}\left[\nabla \mathbf{v} + (\nabla \mathbf{v})^T\right] \tag{4}$$

is the rate-of-strain tensor. For Newtonian fluids this constitutive equation is usually substituted directly into the momentum balance Eq. 2, but for viscoelastic constitutive equations it is expedient to solve the constitutive equation separately with a mixed method, as discussed below.

Typical boundary conditions for the fluid momentum equations can be expressed in terms of surface normal $\mathbf{n}$ and surface tangent $\mathbf{t}$ vectors. The most common boundary conditions are those that prescribe no-slip/impenetrability, applied as usual at fluid-solid surfaces,

$$\mathbf{n} \cdot \mathbf{v} = v_{\mathrm{n}} \qquad \text{and} \qquad \mathbf{t} \cdot \mathbf{v} = v_{\mathrm{t}} \quad . \tag{5}$$

v_n and v_t are the normal and tangential velocity respectively. The natural boundary conditions or zero stress condition, i.e.,

$$\mathbf{nn} : \mathbf{T} = 0 \qquad \text{together with} \qquad \mathbf{t} \cdot \mathbf{v} = 0 \tag{6}$$

is usually applied at fully developed outflow/inflow boundaries in lieu of a specified flow rate. For viscoelastic flow problems, however, an outflow velocity boundary condition must also be specified. Inflow/outflow velocity conditions of the form

$$\mathbf{v}|_\Gamma = \mathbf{g}(x, y, z) \tag{7}$$

are used when the velocity field $\mathbf{g}(x, y, z)$ is prescribed along the boundary Γ.

It is noteworthy in this system of equations that the unknown, dependent variables include all components of the velocity vector and the pressure. The continuity equation (Eq. 1) is generally responsible for the pressure, yet pressure is absent from this equation. The ramifications of this peculiar trait with regard to the theme of this paper are discussed in section 4.

For moving-mesh problems we solve Eqs. 1 and 2 together with the equations governing mesh motion, or position. Here we employ the differential-equation-based approach described by Sackinger et al. [53] in which the static momentum balance of a neo-Hookean nonlinear elastic material is solved for the mesh displacement components. This balance takes on the following form for an inertialess, body-force-free solid material:

$$\nabla \cdot \mathcal{T} = 0 \quad . \tag{8}$$

The stress tensor $\mathcal{T}$ is related to the deformation by an appropriate constitutive relation. For purposes of this discussion, we will use the following nonlinear-elastic, neo-Hookean constitutive form,

$$\mathcal{T} = -P\mathbf{I} + 2\mathrm{G}\mathbf{E_m} \tag{9}$$

where P is the pressure in the solid, $\mathbf{I}$ is the identity tensor, G is the shear modulus and $\mathbf{E_m}$ is the deviatoric portion of the strain tensor defined by the Lagrangian deformation gradient tensor $\mathbf{F_m}$, viz.,

$$\mathbf{E_m} = \frac{1}{2}(\mathbf{F_m^T F_m} - \mathbf{I}) \quad . \tag{10}$$

$\mathbf{F_m}$ is defined as

$$\mathbf{F_m} = \frac{\partial \mathbf{x_m}}{\partial \mathbf{X}} \tag{11}$$

where $\mathbf{x_m}$ is the vector of deformed mesh coordinates. $\mathbf{x_m}$ is related to the stress-free state coordinates $\mathbf{X}$ and the local displacements $\mathbf{d}$ by $\mathbf{x_m} = \mathbf{X} + \mathbf{d}$. We have written the solid constitutive equation in an incompressible form, which introduces a solid pressure. Nothing significant in this formulation changes when the compressible form is invoked, except that there is no longer any need to supplement the system with an incompressibility constraint in the solid. In the examples below we use the compressible form in regions of arbitrary mesh motion in the fluid phase (viz., independent of the material) and the incompressible form in our computational Lagrangian solid phase regions.

When the solid phase is incompressible we must solve an additional continuity equation for the pressure in the solid:

$$\det \mathbf{F_m} = 1 \quad . \tag{12}$$

It is a simple matter to show that this continuity equation is kinematically and thus mathematically equivalent to the fluid-phase continuity equation (Eq. 1) [2] (see Marsden and Hughes [44]). Moreover, as in the solenoidal velocity constraint for incompressible fluids, we again have a situation where the pressure for which this equation accounts is not actually present. This again has clear implications to the numerical methods employed, cf. Section 4.

In ALE regions of the mesh, displacements emanate from free-surface motion, which in turn are governed by so-called distinguished boundary conditions. These boundary conditions connect the grid motion to the problem dynamics or kinematics and are applied to the mesh-motion/position equations. One such condition prevalent in most free and moving boundary problems is the kinematic boundary condition:

$$\mathbf{n} \cdot (\mathbf{v} - \mathbf{v}_s) = 0 \, , \tag{13}$$

written here for a true material boundary across which we have no significant transfer of mass (e.g. by evaporation, condensation, etc.).

Applying Eq. 13 to the mesh-motion equations is tantamount to forcing the free surface to be a material surface, thus defining the extent of the material and mesh. Note that even though this condition contains geometrical information embodied in the normal vector $\mathbf{n}$ and the mesh velocity $\mathbf{v_s}$, the dominant variable contribution is derived from the fluid velocity. Because this is a boundary condition on mesh displacement or position, the discretized condition can generate weak contributions on and near the diagonal of the matrix, especially if $\mathbf{v_s}$ is small and the surface curvature is small. Similar effects from other types of distinguishing condition can be even more detrimental to the matrix system. In phase-change problems it is often desirable to force the mesh to conform with an isotherm or isoconcentration surface. Such conditions contain absolutely no geometric information and hence produce zeros on the diagonal of the resulting matrix system when applied to distinguish mesh motion along a boundary (Sackinger et al. [53]). Clearly, in addition to the incompressibility constraints, boundary conditions can also lead to poor matrix conditioning, as our free surface example in Section 6 reveals.

We consider two types of moving boundaries, one between two fluids, usually a liquid and a gas, and the second between a fluid and a solid. Both types define the extent of a material, and hence are material surfaces. Accordingly, Eq. 13 is used as a boundary condition on the mesh motion/position. The forces along the boundary must balance because the interfaces are presumed to have no mass; they are mathematical lines or surfaces in space. For liquid/gas surfaces characterized by significant surface tension forces, we enforce a balance of viscous stress and capillary

pressure,

$$\mathbf{n} \cdot \mathbf{T} = \mathbf{n} \cdot \mathbf{T_{gas}} + \mathbf{2}\mathcal{H}\sigma\mathbf{n} \tag{14}$$

as a boundary condition on fluid momentum. Here $2\mathcal{H}$ is the mean curvature of the interface and σ is the surface tension. At fluid-solid interfaces sans shell or curvature-dependent forces the condition is similar:

$$\mathbf{n} \cdot \mathbf{T}_{\text{solid}} = \mathbf{n} \cdot \mathbf{T}_{\text{liquid}} \quad . \tag{15}$$

In a later example problem we augment the fluid continuity and momentum relation with an equation that accounts for viscoelastic fluid behavior. Correspondingly we consider a stress tensor of the form

$$\mathbf{T} = -p\mathbf{I} + \tau + 2\mu\mathbf{D}, \tag{16}$$

where the extra stress tensor τ is composed of k modes,

$$\tau = \sum_{k}^{m} \tau_k \quad . \tag{17}$$

Along with Eqs. 1 and 2 we solve a multimode Giesekus model for the stress constitutive equation

$$\frac{\alpha_k \lambda_k}{\eta_k} \tau_k \cdot \tau_k + \tau_k + \lambda_k \overset{\Delta}{\tau}_k = 2\eta_k \mathbf{D} \tag{18}$$

where $\overset{\Delta}{\tau}_k$ is the upper convected derivative of the extra stress tensor, λ_k is a relaxation time, α_k is a mobility factor, and η_k is the polymer viscosity. In addition to the stress equation, our formulation requires an equation for continuous velocity gradient, G, be solved

$$G = \nabla \mathbf{v} \quad . \tag{19}$$

The hyperbolic nature of Eq. 18 complicates many Galerkin-based discretization schemes and has forced many to abandon the standard GFEM approach to have any hope of achieving stable solutions for high elasticity. We address the problem by following the the Elastic Viscous Stress Splitting (EVSS) developed by Rajagopalan et al. [46] and simplified by Guenette and Fortin [30]. Following Fortin and Fortin [26], we use a discontinuous Galerkin formulation to handle the hyperbolic nature of the stress constitutive equation [3]. Because the stress cannot be substituted into the momentum equation and eliminated, as in the Newtonian case, we must solve an additional tensor equation for the stress.

In summary, we take four additional measures to ensure convergence over a maximum range of elasticity: (1) reformulating the momentum equation to make explicit the elliptic character with respect to the velocity field, (2) applying streamline-upwind/Petrov-Galerkin formulation to the stress equation to help stabilize it at

high fluid elasticity (best measured by the Deborah number which is the ratio of a relaxation time λ to a characteristic flow-problem time, t), (3) introducing additional definition equations for the velocity gradient tensor to allow for smooth tensor values at node points, and (4) applying a discontinuous Galerkin formulation to the stress equation. One strategy that we do not pursue, although it is common, is to segregate the stress equation from the coupled Newton algorithm.

The mathematical statement of the problem above and how its various features impact and control numerical solution difficulty depends upon the form of the equations solved, the discretization approach, and the nonlinear iteration strategy chosen. Here we focus on a common approach that begins with the differential equations and boundary conditions in their primitive variable form, i.e. in terms of velocity, pressure and displacements. If we are solving the viscoelastic equations, we must also include the stress and velocity gradient tensors. We solve directly for the velocity components, the pressure, and any other auxiliary variable arising from additional mechanics or transport (e.g. displacements from moving-mesh problems or deformable solid regions and temperatures from non-isothermal problems). We solve all the equations implicitly and simultaneously, which means that when transformed into a matrix system, the problem physics are embodied in a single monolithic matrix. Benefits and drawbacks to common alternative forms (e.g. pressure-Poisson's form, stream function/vorticity form etc.) are discussed in Section 4. In our class of applications this fully-coupled, primitive-variable approach has proved to be the most reliable and extensible.

The application of GFEM to the foregoing equations can be summarized as follows: (1) the physical domain is broken up into elements, (2) a set of finite element basis functions is defined for each element so that functional representations of the independent variables can be constructed (these basis functions are nearly orthogonal, linearly independent polynomials of low-order), and (3) each differential equation is written as a residual, i.e., all terms are put on one side, multiplied by the appropriate basis function in each element and integrated over the entire domain. The resulting expressions are actually termed weighted residuals, denoted here as $\mathbf{R}$. Details for general applications of this scheme are fully described elsewhere [47].

Central to this paper is treatment of the continuity equation. For incompressible flows this equation is used to account for the unknown pressure in the momentum equations. A functional representation of that pressure and the velocity components are constructed with appropriately chosen basis functions, ψ and ϕ viz.

$$p = \sum_{i}^{N^e} \psi_i p_i \qquad \text{and} \qquad v = \sum_{i}^{N^e} \phi_i v_i \tag{20}$$

where N^e are the element functional nodes. Here ψ_i and ϕ_i are suitably chosen low-order polynomials. The standard Galerkin approach necessitates an interpolation of both fields satisfying the classical $inf - sup$ or Babuska-Brezzi stability condition [7]. This compatibility condition is known as the LBB-condition. In this work ψ_i are chosen to be lower order than ϕ_i. We typically choose quadratic functions (biquadratic for two dimensions and triquadratic for three dimensions) for ϕ_i and

linear polynomials for ψ_i This is denoted hereafter as the Q2Q1 element type. We also on occasion will use linear basis functions for velocity and piecewise constant functions for pressure, denoted hereafter as Q1P0 element types. The two LBB-compliant element types used in this work are diagrammed in Figure 1 for both two-dimensional and three-dimensional flow (cf. [7]). With regard to the continuity

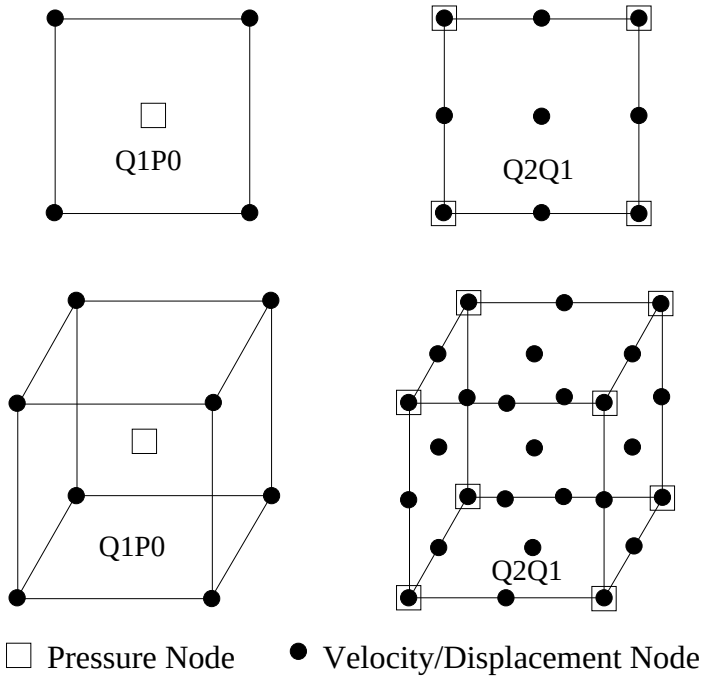

Figure 1: Typical FEM velocity/displacement-pressure mixed elements.

equation, we apply a weighting function identical to the basis function used to represent the pressure, as required by the Galerkin approach:

$$R_c = \int_D \psi \ (\nabla \cdot v) \ \ dV \tag{21}$$

For purposes of this discussion we will choose the same, low-order polynomial representation for the displacements, viscoelastic stress and velocity gradient tensor. However, for the stress, we employ discontinuous versions of the low order polynomials.

$$\mathbf{d} = \sum_{i=1}^{N^e} \phi_i \mathbf{d}_i \qquad \tau_k = \sum_{i=1}^{N^e} \phi_i^d \tau_{(k)\ i} \qquad G = \sum_{i=1}^{N^e} \phi_i G_i \quad . \tag{22}$$

In accordance with Galerkin's method, we form weighted residuals $\mathbf{R}$ of the fluid momentum equation (Eq. 2), the solid momentum equation Eq.(8) and velocity gradient equation (Eq. 18) with the same functions ϕ_i and discontinuous ϕ_i^d for the viscoelastic stress equation. Because we use the same level of basis function for

displacements and mesh coordinates, the associated integration is facilitated by an isoparametric mapping.

The foregoing operations yield a residual vector of nonlinear algebraic equations that we denote as

$$\mathbf{R^T}(\mathbf{v}, \dot{\mathbf{v}}, \mathbf{p}, \mathbf{d}, \mathbf{P}, \tau) = [\ \mathbf{R_{fm}}, \mathbf{R_c}, \mathbf{R_{sm}}, \mathbf{R_{dc}}, \mathbf{R_{sc}}, \mathbf{R}_\tau, \mathbf{R_G}]^\mathbf{T} = \mathbf{0^T} \quad . \tag{23}$$

where $\mathbf{0^T}$ is a column vector of zeros. For purpose of discussion we have broken the residual equation set into several relevant parts: $\mathbf{R_{fm}}$ refers to the residual vector of the fluid momentum equations, $\mathbf{R_c}$ the fluid continuity equation, $\mathbf{R_{sm}}$ the solid momentum equations; $\mathbf{R_{dc}}$ the mesh distinguishing conditions, $\mathbf{R_{sc}}$ the mesh/solid continuity equation, $\mathbf{R}_\tau$ the viscoelastic stress equations, and $\mathbf{R_G}$ is the velocity gradient equation. With one exception these residual elements represent the basic classes of physics. We single out distinguishing boundary conditions on mesh motion, $\mathbf{R_{dc}}$, as they are exemplary of the poor conditioning that can result from implicitly defined boundary conditions. Several remarks are relevant here, as they pertain to the matrix solution algorithms to follow:

- In practice this system of equations is not partitioned in the order shown unless forced by the presence of different material regions in which different mechanics equations are being solved. This representation is put forth for convenience. For computational expedience the bandwidth of the matrix is minimized by alternating the residuals of the different physics classes at each node.

- No restrictions are placed on which elements are active in various regions of the domain. Indeed we can have fluid materials with viscoelastic stresses abutting solid materials undergoing deformation. In this case the collated representation is more appropriate as those associated with each material block are typically contiguous in storage.

- This system of nonlinear equations can be large, (of the order of 10,000 to 100,000 unknowns in this work) and representative of a two-dimensional or three-dimensional problem.

A solution of the discretized equations is obtained by first linearizing the residuals, if necessary, by means of Newton's method or a quasi-Newton variant. Linearization by Newton's method leads to the solution of the matrix equation:

$$J(\mathbf{u})\Delta\mathbf{u} = -R(\mathbf{u}) \tag{24}$$

where $J(\mathbf{u})$ is the so-called Jacobian matrix, $\Delta\mathbf{u}$ is the vector of nodal degree of freedom updates, $R(\mathbf{u_n})$ is the residual vector and $\mathbf{u}$ is the vector of unknown nodal degrees of freedom denoted as $\mathbf{u} = [\ \mathbf{v}\ , p\ , \mathbf{d}\ , P\ , \tau\ ]^T$. The basic collated structure of $J(\mathbf{u})$ is shown in Figure 2. This matrix represents the sensitivity of the residual vector with respect to all degrees of freedom in the problem.
The characteristics of this matrix that motivate this work are the following:

	$\mathbf{v}$	p_{fluid}	$\mathbf{d}$	P_{solid}	τ	G	Misc.
Fluid Momentum	$J_{\text{fm,v}}$	$J_{\text{fm},p}$	$J_{\text{fm,d}}$	0	$J_{\text{fm},\tau}$	$J_{\text{fm},G}$	
Fluid Continuity	$J_{\text{c,v}}$	0	$J_{\text{c,d}}$	0	0	0	
Solid Momentum	0	0	$J_{\text{sm,d}}$	$J_{\text{sm},P}$	0	0	
Mesh DC's	$J_{\text{dc,v}}$	$J_{\text{dc},p}$	$J_{\text{dc,d}}$	0	0	0	
Solid Continuity	0	0	$J_{\text{sc,d}}$	0	0	0	
Viscoelastic Stress	$J_{\tau,\text{v}}$	0	$J_{\tau,\text{d}}$	0	$J_{\tau,\tau}$	$J_{\tau,G}$	
Velocity Gradient	$J_{G,\text{v}}$	0	0	0	0	$J_{G,G}$	
Misc.							

Figure 2: FEM J Matrix.

1) It is generally large and sparse. Although solvable by direct methods, its bandwidth typically prohibits such approaches for three-dimensional problems.

2) All blocks in J are generally nonsymmetric. The mere presence of nonlinear advective terms makes $J_{fm,v}$ and $J_{\tau,v}$ nonsymmetric; the mixed method and Galerkin's approach even makes $J_{c,v}$ nonsymmetric. Perhaps most detrimental to the conditioning of J, outside the zero block along the diagonal, are that dispersed throughout are boundary condition equations, like the distinguishing conditions that give rise to further asymmetries.

3) $J_{c,v} = J_{fm,p}^T$ that arise from the continuity equation and the pressure sensitivity of the fluid momentum equations are related. But, because the continuity equation for the fluid has no contribution from the hydrodynamic pressure, a conspicuous zero matrix block results on the diagonal. This trait alone is the single-most reason why this matrix system has challenged linear-algebraists for years. It leads to matrix indefiniteness, or unbounded eigenvalues of the systems. In other words, any $\mathbf{LD^TL}$ decomposition of the composite matrix will require pivoting of some sort (Golub and Van Loan [28]).

4) $J_{dc,d}$, or the sensitivity of the mesh motion distinguishing conditions for ALE regions of the domain, generally exhibit weak sensitivities relative to other degrees of freedom. Because these boundary conditions are applied to the mesh solid momentum equations, small or negligible entries near the diagonal of J result. This exacerbates the poor conditioning.

5) Letting $*$ denote the complete set of unknowns in the matrix row then the only scaling connection between the fluids blocks $J_{fm,*}$ and the solid deformation blocks $J_{sm,*}$ is the fluid-solid stress balances. In general, these these matrix entries possess a drastically different scale, which can exacerbate the poor conditioning.

6) $J_{\tau,*}$ sub-matrices can possibly be of a greatly different scale than those associ-

ated with the fluid momentum and continuity equations. This is especially true when fluid elasticity becomes more important as the velocity components, the pressure and stress components are different orders of magnitude, leading to a poorly-conditioned nonlinear matrix system. This is especially true in the vicinity of singularities, where the stress components can grow unbounded and the velocity components do not vary dramatically and remain small.

7) Finally, the miscellaneous sub-blocks are included to indicate that we are generally dealing with additional boundary conditions and augmenting conditions that either result in or exacerbate detrimental traits in the Jacobian matrix, like increasing bandwidth, adding more zeros to the diagonal, etc.

The most robust iterative methods have historically performed poorly on matrix systems generated with the strategy discussed above. Even in the simplest case we address below, incompressible fluid mechanics with a mixed velocity-pressure ($\mathrm{v}-p$) formulation, matrices are indefinite and poorly conditioned (Engelman and Hasbani [22]) often causing convergence that is erratic or distressingly slow. Achieving robust convergence properties for general Navier-Stokes problems remains one of the outstanding challenges in numerical analysis.

4 Historical Review

Over the years, numerous techniques have been applied to solving matrix systems of this sort (cf. Eq. 24); in fact, there are far too many to cover in this paper. The review in this section will focus on the basic tactics developed at least in part to mitigate the task. We focus our review on incompressible Navier-Stokes equations, but also look at viscoelastic equations. To date we know of no work involving fluid-structural interactions in the framework it is addressed here. It is noteworthy that the historical factors motivating the four expedients presented in this section are not solely related to the heretofore lackluster and unreliable performance of iterative methods. Other factors have also played a role, including computer memory limitations, floating-point operation count reduction, wider range of convergence of the nonlinear iteration scheme, time-stepping flexibility, and basic ease of programming.

4.1 Pseudo compressibility formulation

The first tactic we discuss is probably the oldest and simplest to implement. Chorin [12] introduced a pseudo-compressibility approach to solving incompressible flow problems. The idea is to apportion the time-derivative of the density to the solenoidal constraint, thereby allowing for some small measure of compressibility,

$$\epsilon \frac{d\rho}{dt} + \nabla \cdot \mathbf{v} = 0. \tag{25}$$

Here ϵ is an adjustable parameter. If the density is taken as a mild function of pressure, discretization of this equation leads to contributions on the diagonal of $\mathbf{J}$, which is otherwise zero for the basic $\mathrm{v} - p$ form advocated here (cf. Figure 2).

The addition of this term offers a dual benefit. First, a time derivative now appears in the continuity equation residual, whereas before it was an algebraic constraint. This allows the equation to be advanced with time integration techniques not subject to the limitations of a differential/algebraic system (DAE, see Gresho [29]). Second, this approach also contributes to the diagonal and hence lowers the condition number of $\mathbf{J}$. These benefits come at a heavy price, however, especially for low-speed flows. Accuracy degradation of the velocity and pressure fields is enormous, especially when ϵ is large enough to lend significant benefit to iterative solver performance. Moreover, near boundaries where no allowance is made for this compressibility effect, significant degradation to mass conservation becomes evident. Finally, this technique does not offer any advantage to solving the steady equations, as it rests on advancing the continuity equation in a time-dependent manner.

4.2 Pressure Penalty Formulation

One can circumvent the steady-problem encumbrance by deploying a so-called pressure penalty approach. The penalty formulation of the continuity equation is

$$\nabla \cdot \mathbf{v} + \frac{p}{\lambda} = 0 \tag{26}$$

Here λ is the penalty parameter: it must be assigned a large value in order to approximate incompressibility. This equation actually describes a field of distributed mass source $\nabla \cdot \mathbf{v}$ and sink p/λ in which the local pressure is proportional to the rate of annihilation of mass (and locally negative pressure is proportional to the rate of mass creation). Despite these deleterious physical traits, when solved with discontinuous linear basis functions for pressure this equation leads to a penalty method that can yield solutions to viscous flows virtually identical to those from the direct mixed interpolation approach advocated in this work (cf. Khesghi and Luskin [40]).

Penalty approaches such as this one were developed early on (e.g. Zienkiewicz [64]) for a variety of reasons, including obvious benefits of ease of programming and reduced problem size. Its main attribute is that it provides a way to decouple the pressure calculation from the velocity (note that Eq. 26 can be used to eliminate the pressure in Eq. 2). Most pertinent to this paper, however, is the work of Engelman and Hasbani [22] who recognized that Eq. 1 was the root cause of poor iterative solver convergence in their quest for more efficient matrix solvers for large problems. As one attempt to circumvent this problem they turned to a pressure penalty formulation, which clearly results in a finite contribution on the diagonal of the matrix equations corresponding to the continuity. Although they found that iterative solver performance improved dramatically, the tradeoff between large condition numbers at large λ and large inaccuracies at smaller λ could not be managed so as to yield a reliable and robust algorithm. To realize maximum solver performance benefit they concluded that too much accuracy had to be compromised. An interesting conclusion of the work of Engelman and Hasbani is motivating here: they declared

that the only viable solution strategy for arbitrarily large problems was to segregate the system. In fact, many researchers have been led to this conclusion; any way to decouple the hydrodynamic pressure from the system allows greater flexibility in solution strategies and allows for the generation of sub-matrices that are smaller and contain entries on the diagonal and hence are more amenable to preconditioned iterative solution techniques (see Gresho [29]) .

4.3 Segregated Methods

Note the matrix system symbolically represented by the set in the previous section (Figure 2) is not immediately amenable to a segregated approach because some blocks do not contain diagonal components, viz. those associated with the continuity equation. The basic idea of segregated algorithms is to decouple the pressure calculation from the velocity calculation by taking the divergence of the vector momentum equation and applying some clever insights regarding incompressible flow. Early on the motivation for this approach was largely two-fold: to mitigate memory requirements of fully-coupled algorithms and to enable semi-implicit time integration.

Pressure/velocity segregation methods have been reviewed by several researchers in the context of the finite element method; most notable are the papers by Gresho [29] and Haroutunian et al. [31]. The origins of pressure/velocity decoupling harks back to several key developments in computational fluid dynamics (CFD) using finite difference and finite volume discretization. In fact, many of the modern algorithms have been derived from the voluminous experience base accumulated by CFD practitioners contending with limited memory capacity in the 1970s and 1980s. Haroutunian et al. [31] review the progression of techniques from early on in the finite volume and finite difference arena and have concluded that the SIMPLE algorithm and its variants (Patankar [45]) have emerged as the most reliable and robust approach. Haroutunian et al. then proposed finite element counterpart to the SIMPLE algorithm applicable to the problem at hand, with special attention given to solving steady-state problems without false-transient time integration.

Basically all current segregated algorithm variants are distinguished by the way in which the pressure is decoupled and projected from one time step to the next. Some approaches lead to a vorticity/velocity equation while others lead to a stream function-vorticity equation (cf. Gresho [29]). In every case one is faced with solving the pressure-Poisson equation of the form

$$\nabla \cdot \nabla P = \nabla \cdot \mathbf{f} - \nabla \cdot (\mathbf{v} \cdot \nabla \mathbf{u}) \tag{27}$$

obtained by taking the divergence of Eq. (2) and applying Eq. (1). Haroutunian et al. [31]) proposed three consistent finite element counterparts to the SIMPLE and SIMPLER algorithm. To further reduce the size of the sub-matrix systems, each individual component of the momentum equations were solved separately and successively by iterative techniques. Overall at each Newton iteration or Picard iteration they solved four matrix subsystems, one for each of three velocity components and one for the pressure.

Interestingly, the most challenging matrix system to solve happens to be one arising from the discretization of the pressure equation (Eq. 27) – here the right-hand-side is lagged from the last iteration so that this equation is solved solely for the pressure. The resulting matrix, despite being symmetric, is actually very poorly conditioned due to poor scaling. Nonetheless, these challenging matrix systems can be readily solved by modern iterative solvers and reordering/preconditioner strategies, and the success of this algorithm over the last decade has been enormous. As a further relevant example to this work see Fortin and Fortin [24] who applied a segregated approach to a viscoelastic flow problem. Problems of substantial size (greater than 1 million unknowns) have been solved routinely and the algorithm has been exploited for its robustness and generality in a variety of widely used commercial codes.

In our view this approach is still a compromise to the the favorable convergence properties of a fully-coupled technique advocated here. Convergence to steady solutions or at successive time steps is linear at best and sometimes even asymptotic, making for sometimes large number of required segregation iterations (albeit fast iterations). Moreover, the method introduces several relaxation parameters that must be tuned to the application. If one chooses alternatives to the primitive variable formulation then boundary conditions on velocity become difficult to apply accurately (cf. the vorticity and stream-function approaches discussed by Gresho [29]). Finally, codes centered around this algorithm are more complex in structure, as they still contain the intricacies of matrix solution services but involve more than one data structure (for the mixed interpolation) and more inner and outer iteration loops. These and other encumbrances has motivated the current authors to employ the same full-Newton, primitive variable, coupled approach that has proved useful for so many two-dimensional applications, in three dimensions.

4.4 Pressure stabilization techniques

As they were developing a segregated Navier-Stokes solver, Haroutunian et al. [31] cite contemporary research in the finite element community towards the development of pressure stabilization schemes. The prime motivation of pressure stabilization is to circumvent the cumbersome LBB constraint on allowable element types. The main goal was to generate a scheme that allows use of the computationally convenient Q1Q1 $\mathrm{v} - p$ element, but many other advantages grew out of this effort. The most well-known scheme is the Pressure Stabilized Petrov-Galerkin (PSPG) approach, although other formulations tantamount to PSPG have appeared (e.g. Mini-Element for Stokes flow by Arnold et al. [2]).

The PSPG method employs standard velocity stabilization of the streamlined upwinding variety for advective terms of the momentum equation and a Galerkin Least Squares formulation of the momentum equation to stabilize the continuity equation (cf. Hughes et al. [37]). The standard finite element formulation weights the momentum equation and the continuity equation with the shape function used to describe the interpolation of the variables. When pressure stabilization is used, the momentum equation is dotted into the gradient of the Galerkin weight function

and added on to the Galerkin continuity equation weighted residual. This leads to a nonzero diagonal and results in a modified form of continuity residual Eq. 21:

$$\begin{aligned} R_c &= \int_D \phi \nabla \cdot \mathbf{v} \; dV \\ &+ \int_D \tau_{pspg} \nabla \phi \cdot \left[\rho \frac{\partial \mathbf{v}}{\partial t} + \rho(\mathbf{v} - \mathbf{v}_s) \cdot \nabla \mathbf{v} - \mathbf{f} - \nabla \cdot \mathbf{T} \right] dV = 0 \quad . \end{aligned} \tag{28}$$

Typically τ_{pspg} is chosen to be of scaling based on the element-average Reynolds number (Re_e) of the flow. Based on suggestions in the literature, we let

$$\tau_{pspg} = \begin{cases} \frac{\alpha h_{elem}^2}{12\mu} & (\mathrm{Re}_e < 3) \\ \\ \frac{\alpha h_{elem}}{2\rho U_{norm}} & \text{otherwise} \end{cases} \tag{29}$$

where α is the pressure-stabilization constant and h_{elem} is the characteristic element length. Notice that the term multiplied by τ_{pspg} (the bracketed term of Eq. 29) is derived from a least-squares residual equation. More importantly, this residual equation exhibits a dependence on pressure, which means that nonzero contributions will result on the diagonal of the Jacobian matrix. This term can be shown to converge with mesh more slowly than the first term, thereby undermining the accuracy (see results below). Moreover, we note that a higher-order derivative of the velocity field is present, viz. $(\nabla \cdot (\nabla v))$ arising from $\nabla \cdot \mathbf{T}$, thereby necessitating a higher-order basis function to evaluate. Most chose to ignore this term in practice (cf. Hughes and Franca [38], etc), even though it was recognized by Droux and Hughes [18] to be essential to the convergence and accuracy of the method at low Reynolds numbers. Droux and Hughes devised boundary integrals that ostensibly shore up the accuracy and convergence for this class of problems.

Use of PSPG schemes and related element-basis-function manipulation techniques are now quite prevalent. Edis and Aslan [19] and Codina and Blasco [16] have used a variant that gives an LBB compliant Q1Q1 element in the context of fractional-step pressure projection methods. Their approach involves a heightened number of velocity degrees of freedom in the element. Howard et al. [35] explored the benefits of PSPG with respect to iterative solver performance, making many comparisons to direct methods for a small class of incompressible flow problems. Tezduyar and his research group have performed simulations of impressive size using this unique space-time finite element formulation with stabilized Q1Q1 elements (cf. Tezduyar et al. [58]).

The current authors have implemented this scheme as stated with the purpose of improving the performance of iterative solvers/preconditioners on our fully-coupled formulations – it is perhaps noteworthy that Hughes et al. [37] cited this as an added benefit. Although this goal was realized, due mainly to the population of the diagonal components of $\mathbf{J}$, we have found several disconcerting byproducts of this approach that precludes its general usage. Two of the test problems in Section 6 were designed to elucidate the poor solution quality that can result with the PSPG

approach as applied to low Reynolds number confined flow. In another test problem with viscoelastic flow we found that the PSPG approach resulted in unacceptable convergence of the matrix solver.

4.5 Fully-coupled, mixed LBB

Strong precedence to this work are a handful of published attempts to accomplish what we intend to here: successfully apply iterative solvers to the LBB-compliant Galerkin finite element formulations of the Navier-Stokes equations. Based on these attempts it has become clear that preconditioning matrix systems is far more effective than choosing a different iterative technique. Caballero and Seider [9] cite more than 80 papers addressing this subject prior to 1999, in the general case. For incompressible flow problems, they found the most popular preconditioners are based on ILU/triangular factorization, consistent with our approach here.

Einset and Jensen [20] with a Q2Q1 element and a fully-coupled Newton's method solved three-dimensional gas flows in Chemical Vapor Deposition reactors using GMRES and a specialized preconditioner that was highly dependent on the mesh structure. They obtained acceptable convergence behavior, but only after invoking equation reordering and pivoting with their incomplete LU preconditioner; this maneuver has strong precedence here. Perhaps most telling is that their largest problems size was 15000 unknowns in three dimensions, and that required the then state-of-the-art CRAY YMP supercomputer.

Salinger et al. [54], with a similar mixed finite element method, had some success by simply modifying the matrix preconditioner such that no zeros occur on the diagonals, viz. $D = J_{ii} + \epsilon$ where ϵ is chosen to avoid dividing by zero on the continuity equations. This approach, albeit simple, seems to work for their low-speed transient gas flow and is advocated here in our block class of preconditioners.

As previously mentioned, Engelman and Hasbani [22] attempted to apply GMRES with a variety of preconditioners to the fully-coupled system, but with the pressure penalty on the continuity equation. Their work, although resulting in the abandonment of the approach advocated here, was comprehensive and outward looking. For instance they make it clear that if success were to be realized, one should really use inexact Newton methods to avoid unnecessary linear solve iterations at early Newton steps (see Shadid et al. [55]). They also note that since indefinite systems contain zero pivots, the incomplete factorization will require some sort of pivoting. Finally, they suggest a QR-algorithm-based preconditioner (cf. Saad [48]), which is similar to the approach we take up below. An interesting feature of their work is that they seem to have tried all of the best available preconditioning techniques. One reason we believe they were unsuccessful is explained in the conclusions of this paper.

Many have made attempts at applying iterative solvers to flows with complex rheological behavior, perhaps because these problems have a relatively greater computational expense than their Newtonian counterpart due to much larger bandwidths. Carey et al. [11] with some success applied conjugate-gradient-squared algorithms to incompressible flow of shear-thinning liquids. They used the classic

finite element mixed formulation and pressure penalty formulation with considerable success for a two-dimensional lid-driven cavity flow. They also explored the bi-conjugate gradient method and GMRES. Most noteworthy is that they used a direct factorization of the equivalent Stokes problem (i.e., Re = 0) as a preconditioner –obviously an impractical tactic for three-dimensional problems. What sets precedence here though is that they actually used a preconditioner that reordered and processed the incompressibility constraints via ILU with pivoting. They also observed a rapid degradation in convergence behavior as the shear-thinning became more severe.

Similar to our interests, Baaijens [4] advocated the Taylor-Hood Q2Q1 element type in his work on incompressible flow of viscoelastic fluids. He also employed a fully-coupled Newton-Raphson based approach. His preconditioning strategy amounts to condensing out the discretized stress and velocity gradient equations, via Schur-complement, and using that as a preconditioning matrix. It is noteworthy that Baaijens advocated the PSPG-related approach in 1995, but is now looking at non-PSPG schemes, which is perhaps related to the limitation of the former.

Prior to Baaijens's work, Tsai et al. [59] compared GMRES, BiCGStab [63], and direct frontal solver for fully-coupled incompressible mixed formulation together with a split stress equation for an Oldroyd-B fluid. His calculations were two-dimensional and of moderate problem size. His preconditioning strategy was basically ILU using frontal elimination algorithm with pivoting. The moderate problem sizes in two dimensions and the lack of follow-on efforts indicate that Tsai and coworkers were limited by memory availability.

Chapman et al. [1] recently have taken a general approach of applying a progression of ILU-based preconditioning strategies. Their target problem class includes matrices arising from the mixed v-p formulation of the incompressible Navier-Stokes equations, much like those targeted here. They solved matrix systems created by FIDAP [39] with considerable success, but concluded that quite high levels of fill-in were required. Strangely they spoke little of the physical and mathematical origins of their target system, examined only two-dimensional problems, and never addressed the limitations of ILU-based preconditioning with problem size growth. In fact, we have found that the fill-in levels they employed for two-dimensional problems are strictly prohibitive in three dimensions, even on modern large-memory computers. They tout block preconditioners but do not test them on indefinite matrices. Other example matrices included some from Harwell-Boeing applications in oil-reservoir modeling and related flows, and BARTH matrices from 2D turbulent flow over airfoils, with a formulation that results in fully-populated matrix diagonals. The only relevant applications here are their tests with the FIDAP matrices and we find their conclusion to be consistent with ours, that ILU with high fill-levels is often required to solve this class of matrices.

Zhang [65] also addresses a host of matrix systems arising from applications in computational fluid mechanics. His goal was to determine experimentally the state of applying BILU preconditioners to general classes of problems. Extensions to the basic block approach that he advocates include singular value decomposition on the block level, with singular-value perturbation in the case of nearly-singular

preconditioning factors, much the same as our approach. His tests are extensive and in all cases he was successful at getting this class of preconditioners to work, but not without effort. He too used matrices generated by the fluid-mechanics finite element code FIDAP [21] and admitted these were the most challenging due to their indefinite nature. Unfortunately little information is given on the prospects of BILU on large three-dimensional problems and no stringent tests on matrices arising from similar formulations, but extended non-Newtonian liquids and fluid-structural interaction problems.

Beyond preconditioning, many practitioners simply resort to transient analysis to overcome poor performance of iterative solvers (e.g. Tezduyar et al. [58], Strigberger et al. [57]); the idea is to take advantage of iterative solvers that thrive on a good initial guess - which transient analysis with small enough time increment delivers at each time step. Furthermore, the smaller the time step the more diagonally dominant the matrix system, because the time derivatives occur on or near the diagonals. Unfortunately transient analysis is inefficient for a broad class of problems that are steady in nature, and hence integrating in time is not the end-all cure.

A final approach worthy of mention are those techniques that seek to avoid forming the matrix altogether, thereby circumventing the "zero-on-the-diagonal" problem with the iterative solver. Most iterative solvers require a matrix vector multiplication, viz. $\mathbf{J}v$. In turn this product can be computed with numerical differencing as $R(\mathbf{u}) - R(\mathbf{u} + \epsilon v)/\epsilon$ where v is a general Krylov vector (see Section 5) and so the method never requires the matrix itself. Several discovered then that Newton's method and the iterative solver can then be woven together, such that the matrix is never formed (Brown and Saad [8]); these techniques are known as Newton-Krylov methods. Although offering many advantages over a matrix-based approach in terms of storage requirements, they are encumbered by poor performance on fully coupled Navier-Stokes systems; the only way to improve matters is by preconditioning, which demands a matrix. Several applications to Navier-Stokes systems with additional coupled physics have appeared in the literature, mostly in the context of finite difference or finite volume formulations and hence are not subject to the same LBB constraints that limit finite element forms (cf. Knoll, Kothe and Lally [41]). A few that involve the finite element method, like work by Fortin and Zine [25] who used a Newton-Krylov approach on the velocity-pressure flow calculation of a viscoelastic fluid. They again deployed some level of segregation, however. Although promising, the matrix-free approach suffers from the same ills as the method advocated here and requires similar sophistication in preconditioning. In fact, Newton-Krylov schemes are usually deployed as a part of a segregated algorithm for most flow problems (cf. Knoll et al. [41]) and have only recently been applied to the fully coupled system. It remains to be seen what additional advantages they will offer to the approach advocated here outside the obvious storage savings.

5 Matrix Solvers

Preconditioned iterative methods have long been popular for solving linear equations that arise in computational fluid dynamics. Early methods include line relaxation schemes for structured grid problems and Gauss-Seidel iterations. Interestingly, both of these classes of methods are still in use today, often as preconditioners for some iterative accelerators such as conjugate gradient methods. As mentioned in Section 4, Haroutunian, Engelman and Hasbani [31] showed that the use of preconditioned conjugate gradient methods, e.g., the symmetric Gauss-Seidel preconditioned conjugate residual method, provide an attractive approach to solving the linear systems that come from segregated finite element approaches. Generally, these methods provide the best overall combination of robustness, ease of use (minimal tuning parameters) and efficient computation and memory use. Typically the nonsymmetric systems of equations, where updates are computed for the velocity components, and perhaps other non-pressure variables, require minimal preconditioning, e.g., diagonal scaling, and one of the many short term recurrence iterative schemes such as Bi-CGSTAB. The pressure-Poisson equation, which is symmetric and positive (semi) definite, usually requires more work and a more robust preconditioner, but is still solvable via the symmetric Gauss-Seidel preconditioned conjugate residual method or related methods.

Unfortunately, the successful use of preconditioned iterative methods for segregated problems has not generally been extended to fully-coupled systems. In fact, one can argue that the lack of affordable preconditioned iterative methods for fully-coupled three-dimensional problems has forced CFD application designers to consider segregated methods even though fully-coupled formulations were otherwise considered superior (as discussed in Section 4). Although two-dimensional problems can be reliably solved by direct methods, and segregated formulations can reliably use iterative methods, fully-coupled three-dimensional problems cannot be solved in a reliable, cost-effective way. Direct methods are far too expensive, if feasible at all, because of memory limitations. Iterative methods have been unreliable.

However, given the availability of large-memory, low-cost computers, and the capability to combine these computers into a single parallel computer, we can now begin to consider the solution of much larger three-dimensional problems. Using incomplete factorization preconditioners and non-restarted GMRES, we are able to reliably solve an increasing number of previously infeasible fully-coupled models.

In the remainder of this section we discuss the details of our incomplete factorization and GMRES implementations, emphasizing the important details that make our implementations more effective than the classic textbook versions of these methods. Throughout this section, we are concerned with solving a linear system of equations

$$Ax = b \tag{30}$$

where A is a square nonsymmetric matrix (e.g. $\mathbf{J}$ in Eq. 24), b is a known right hand side vector and x is the unknown vector we wish to compute.

1. Given a matrix A and right hand side b, choose an initial guess x_0.
2. Compute $r_0 = b - Ax_0, \beta = \|r_0\|_2$.
3. Apply the Arnoldi process to produce an upper Hessenberg matrix $\bar{H}_m$ and orthogonal matrix V_{m+1} such that $AV_m = V_{m+1}\bar{H}_m$ (where V_m is the first m columns of V_{m+1}).
4. Define the new approximate solution $x_m = x_0 + V_m y_m$ where $y_m = \text{argmin}_y \|\beta e_1 - \bar{H}_m y\|_2$, with $e_1 = (1, 0, \ldots, 0)^T$.

Figure 3: Basic GMRES Algorithm

5.1 Iterative Methods

Unlike the case for symmetric matrices, there is no iterative method for nonsymmetric matrices that is simultaneously inexpensive and optimal. In fact, the Faber-Manteuffel theorem [23] states that no iterative method can both minimize the error and have fixed computational cost per iteration. Thus, for fully-coupled systems, we are forced to consider either inexpensive non-optimal methods that have a low fixed cost per iteration or optimal methods with increasing cost per iteration.

Non-optimal methods such as Bi-CGSTAB are attractive to many people because they are easy to implement, highly parallel and inexpensive to use, even if hundreds of iterations are required. However, our experience shows that these methods are usually unreliable for our most difficult problems.

GMRES, an optimal method if no restarting is performed, is more expensive than Bi-CGSTAB, especially if many iterations are required, but is far more robust for our difficult problems. Because of its robustness, and the increasing speed and memory size of modern computers, we have come to rely on non-restarted GMRES as our primary iterative method.

5.1.1 Details of GMRES Implementation

The Generalized Minimum Residual Method (GMRES) is a popular iterative technique for finding solutions to nonsymmetric linear systems of equations. The basic algorithm is described in Figure 3. One of the important implementation details of GMRES is how to perform orthogonalization in the Arnoldi process. Traditionally a modified Gram-Schmidt (MGS) algorithm has been used because it has adequate numerical accuracy and reasonable cost. Householder orthogonalization (HO) can be used to improve numerical accuracy even more, but it costs about twice as much as MGS.

Classical Gram-Schmidt (CGS) is attractive in a parallel computing environment because it has better communication complexity compared to MGS, but its numerical accuracy is often inadequate due to severe cancellation. This problem

can be addressed by performing a second orthogonalization step. In our experience, CGS with this double orthogonalization provides the best combination of parallel efficiency and numerical accuracy. In all the experiments performed below, we use this approach. A detailed description of the orthogonalization options is given by Saad [51].

5.2 Preconditioners

The choice of preconditioner for iterative solution of difficult problems remains a challenge. Major categories of general-purpose preconditioners include diagonal scalings, approximate inverses, algebraic multi-level methods and incomplete factorizations. In this section we briefly discuss the first three categories and then focus on our approach to using incomplete factorizations.

5.2.1 Diagonal Scalings

A formal description of diagonal scaling involves defining two diagonal matrices (or more generally block diagonal matrices) D_L and D_R, one of which could be the identity matrix, and replacing our original linear system 30 with the following:

$$\tilde{A}\tilde{x} = \tilde{b} \tag{31}$$

where $\tilde{A} = D_L^{-1} A D_R^{-1}$, $\tilde{x} = D_R x$ and $\tilde{b} = D_L^{-1} b$. Note that we need not explicitly form $\tilde{A}$, $\tilde{x}$ and $\tilde{b}$. This is an important point for some classes of problems.

Diagonal scaling is very important for problems with large variation in matrix coefficient magnitudes. It should always be considered in combination with some other form of preconditioning and can sometimes be sufficient by itself. A thorough discussion of choices for D_L and D_R is beyond the scope of this paper. For the test problems presented here, we use row sum scaling such that D_R is the identity matrix and $(D_L)_{ii} = \sum_{j=1}^{n} |a_{ij}|$ where a_{ij} denote the matrix coefficients of A. In exact arithmetic, this scaling causes the ∞-norm of $\tilde{A}$ to be 1. We are experimenting with column and two-sided scaling strategies but our results are incomplete at this time.

5.2.2 Other Preconditioners

Research in approximate inverse preconditioners is very active [5, 14, 42] and is producing increasingly more robust strategies. However, our experience has shown that they continue to be less robust than incomplete factorizations. Research in multi-level preconditioners, usually some generalization of classical multigrid approaches [60], is receiving renewed interest as part of the pursuit for preconditioners that scale well on massively parallel computers for unstructured problems. Finally, a new set of techniques [56] that exploits the basic structure of the Navier-Stokes equations is of great interest. All of these techniques promise to advance solver capabilities in the near future.

5.2.3 Incomplete Factorization Preconditioners

One of the most reliable and versatile classes of preconditioners is incomplete factorization. Methods in this class compute an incomplete lower/upper (ILU) factorization of the matrix A in Equation 30, or the scaled matrix $\tilde{A}$ in Equation 31. Although there are still problems we cannot solve, we have found this class of methods to be most suitable for the problems presented in the paper.

There are many variations on incomplete factorizations. Two of the most common subclasses are pattern-based and threshold-based. A good overview of these methods is given by Saad [51]. We focus on two specific types, namely a pattern-based block ILU denoted BILU(k) [15, 33] and a variant of ILUT [49, 33]. The results in this paper are based on using these two preconditioners.

BILU(k) is a patterned based incomplete factorization with which each entry in the sparse matrix is a (small) dense matrix. This type of preconditioning is natural for mesh-based applications where there are multiple degrees of freedom per mesh node. The matrix connectivity corresponds to mesh node connectivity and the dense matrices are the interactions between the degrees of freedom at a mesh node. The level fill parameter k refers to the pattern of the BILU factors. Level fill of k = 0 means that the only block entries kept are those that correspond to the pattern of the original matrix. Level fill of k > 0 is defined recursively to allow the BILU factors to have entries that correspond to the terms from level fill k-1 and any fill-in that comes directly from level k-1 terms.

ILUT is a dual-threshold based preconditioner where factors are computed row by row, dropping terms that fall below the threshold parameter multiplied by the norm of the row, and then keeping (at most) a prescribed number of the largest terms in the row. Details of BILU(k) and ILUT preconditioners can be found in the Aztec User Guide [61].

5.2.4 Diagonal Perturbations

The matrices generated by our test problems are composed of several different types of physics as displayed in Figure 2, and we are simultaneously resolving all degrees of freedom. As a result, our matrix structure can have many forms depending on the particular problem being solved and physics being modeled. One outcome of this is that incomplete factorizations can be difficult to compute, even if the original matrix A is well-conditioned. A few sources of difficulty are:

1. Zero diagonal entries. In this case, unless fill-in occurs prior to dividing by the zero diagonal, or we perform some type of pivoting, the factorization will fail or produce unusable factors. In some instances even when fill-in does occur, the diagonal value may be too small to produce a usable factorization.

2. Singular principle sub-matrices. In this case, boundary conditions are missing or insufficient to determine a nonsingular upper left sub-matrix. Thus any attempt to compute a block ILU factorization will fail.

3. Singularity due to domain partitioning. When executing in parallel using additive Schwarz methods, we observe situations where an incomplete factorization for the entire domain exists but one or more factorizations for the subdomains do not.

One straightforward technique to address poorly conditioned factors is to introduce diagonal, or block diagonal, perturbations. In this situation, the incomplete factorization is performed on a matrix that is identical to A except that diagonal (or block diagonal) entries are perturbed, usually to increase diagonal or block diagonal dominance. This idea was introduced by Manteuffel [43] as a means for computing incomplete Cholesky decompositions for symmetric positive definite systems and extended to nonsymmetric matrices by van der Vorst [62], Saad [50] and Chow [13]. It is used for block entry matrices in a package called BPKIT [15].

Since Krylov methods such as GMRES are invariant under scaling, and a very large diagonal perturbation essentially makes the off-diagonal elements irrelevant, one way to view diagonal perturbation is as establishing a continuum between an accurate but poorly conditioned incomplete factorization and less accurate but perfectly conditioned Jacobi diagonal scaling. Given this continuum, the strategy is then to choose a minimal perturbation that sufficiently stabilizes the factorization.

5.3 Perturbation Strategies

As mentioned above, we often have difficulty computing usable incomplete factorizations for our problems. The most common source of problems is that the factorization may encounter a small or zero pivot, in which case the factorization can fail, or even if the factorization succeeds, the factors may be so poorly conditioned that use of them in the iterative phase produces meaningless results. Before we can fix this problem, we must be able to detect it. To this end, we use a simple but effective condition number estimate for $(LU)^{-1}$.

5.3.1 Estimating Preconditioner Condition Numbers

The condition of a matrix B, called $cond_p(B)$, is defined as $cond_p(B) = \|B\|_p \|B^{-1}\|_p$ in some appropriate norm p. $cond_p(B)$ gives some indication of how many accurate floating point digits can be expected from operations involving the matrix and its inverse. A condition number approaching the accuracy of a given floating point number system, about 15 decimal digits in IEEE double precision, means that any results involving B or B^{-1} may be meaningless.

The ∞-norm of a vector y is defined as the maximum of the absolute values of the vector entries, and the ∞-norm of a matrix C is defined as $\|C\|_\infty = \max_{\|y\|_\infty=1} \|Cy\|_\infty$. A crude lower bound for the $cond_\infty(C)$ is $\|C^{-1}e\|_\infty$ where $e = (1, 1, \ldots, 1)^T$. It is a lower bound because $cond_\infty(C) = \|C\|_\infty \|C^{-1}\|_\infty \geq \|C^{-1}\|_\infty \geq |C^{-1}e\|_\infty$.

For our purposes, we want to estimate $cond_\infty(LU)$, where L and U are our incomplete factors. Chow [13] demonstrates that $\|(LU)^{-1}e\|_\infty$ provides an effective

estimate for $cond_\infty(LU)$. Furthermore, since finding z such that $LUz = y$ is a basic kernel for applying the preconditioner, computing this estimate of $cond_\infty(LU)$ is performed by setting $y = e$, calling the solve kernel to compute z and then computing $\|z\|_\infty$. The condition number estimates reported in Section 6 are obtained using this approach.

5.3.2 *A priori* Diagonal Perturbations

Given the above method to estimate the conditioning of the incomplete factors, if we detect that our factorization is too ill-conditioned we can improve the conditioning by perturbing the matrix diagonal and restarting the factorization using this more diagonally dominant matrix. In order to apply perturbation, prior to starting the factorization, we compute a diagonal perturbation of our matrix A in Eq. 30 and perform the factorization on this perturbed matrix. The overhead cost of perturbing the diagonal is minimal since the first step in computing the incomplete factors is to copy the matrix A into the memory space for the incomplete factors. We simply compute the perturbed diagonal at this point. The actual perturbation values we use are discussed below.

5.3.3 Dynamic Diagonal Perturbations

Another approach to stabilizing the factorization is to modify diagonal values as the factorization is being computed, making sure the the diagonal pivots do not become too small. For scalar diagonal entries, we have not found this approach to be useful. Even though we can ensure the diagonal values remain above some threshold, in practice this does not prevent the factorization from becoming too ill-conditioned.

However, with block-entry matrices, where the diagonals are dense matrices, it has been fruitful to consider dynamic perturbations. In this situation, as we compute the factorization and prepare to apply the inverse of a block diagonal entry, we perform a singular value decomposition (SVD) on the block diagonal entry and replace any small singular values with a threshold value [15]. We then construct the inverse of the block diagonal entry using the modified singular values.

5.4 Strategies for Managing Preconditioner Condition Numbers

Without any prior knowledge of a problem, the first step to take when computing a preconditioner is to compute the original factors without any diagonal perturbation. This usually gives the most accurate factorization and, if the condition estimate of the factors is not too big, will lead to the best convergence. If the condition estimate of the original factors is larger than machine precision, say greater than 1.0e15, then it is possible that the factorization will destroy convergence of the iterative solver. This will be evident if the iterative solver starts to diverge, stagnates, or aborts because it detects ill-conditioning. In these cases, diagonal perturbations may be effective. If the condition estimate of the preconditioner is well below machine

1. Set the absolute threshold $\alpha = 0.0$ and the relative threshold $\rho = 1.0$ (equivalent to no perturbation).
2. Define perturbed diagonal entries as $d_i = sign(d_i)\alpha + d_i\rho$ and compute the incomplete factors L and U.
3. Compute $condest = \|(LU)^{-1}e\|_\infty$ where $e = (1, 1, \ldots, 1)^T$.
4. If failure ($condest > 10^{15}$ or convergence is poor), set $\alpha = 10^{-5}$, $\rho = 1.0$. Repeat Steps 2 and 3.
5. If failure, set $\alpha = 10^{-5}$, $\rho = 1.01$. Repeat Steps 2 and 3.
6. If failure, set $\alpha = 10^{-2}$, $\rho = 1.0$. Repeat Steps 2 and 3.
7. If failure, set $\alpha = 10^{-2}$, $\rho = 1.01$. Repeat Steps 2 and 3.
8. If still failing, continue alternate increases in the two threshold values.

Figure 4: Simple *a priori* Threshold Strategy

precision (less than 1.0e13) and one is not achieving convergence, then diagonal perturbation will probably not be useful. Instead, one should try to construct a more accurate factorization by increasing fill.

5.4.1 Strategies for *a priori* Diagonal Perturbations

The goal when applying *a priori* perturbations is to find a close to minimal perturbation that reduces the condition estimate below machine precision (roughly 1.0e16). For the results presented in Section 6 we use the strategy outlined in Figure 4. Essentially, we replace the diagonal values $(d_1, d_2, \ldots, d_n)$ with $d_i = sign(d_i)\alpha + d_i\rho$, $i = 1, 2, \ldots, n$, where n is the matrix dimension and $sign(d_i)$ returns the sign of the diagonal entry. This has the effect of forcing the diagonal values to have minimal magnitude of α and to increase each by an amount proportional to ρ, and still keep the sign of the original diagonal entry.

5.4.2 Strategies for Dynamic Block Diagonal Perturbations

The same general goal (of *a priori* perturbations) is valid for dynamic block diagonal perturbations. Specifically, we want to choose values of α and ρ that make minimal perturbations, if any, to the block diagonal values. For dynamic perturbations, we have the same α and ρ parameters, but the meaning of the parameters is slightly different and corresponds to the parameters used in BPKIT [15]. For the results presented in Section 6 we use the strategy outlined in Figure 5. This definition of α and ρ guarantees that the singular values of each diagonal pivot block will not be

1. Set the absolute threshold $\alpha = 0.0$ and the relative threshold $\rho = 0.0$ (equivalent to no perturbation).

2. For each block row:

   (a) Compute the singular value decomposition of the i^{th} diagonal pivot block, $D_i = U\Sigma V^T$, where $\Sigma = diag(\sigma_1, \sigma_2, \ldots, \sigma_s)$ is a diagonal matrix of the singular values of D_i in decreasing order [28], and s is the dimension of D_i.

   (b) Redefine Σ such that $\sigma_k = \max(\sigma_k, \alpha + \rho * \sigma_1)$, $k = 1, \ldots, s$.

   (c) Construct $D_i^{-1} = V\Sigma^{-1}U^T$, using the perturbed Σ.

   (d) Proceed with factorization for i^{th} row.

3. If failure ($condest > 10^{15}$ or convergence is poor), set $\alpha = \rho = 10^{-14}$. Repeat Step 2.

4. If failure, set $\alpha = \rho = 10^{-3}$. Repeat Step 2.

5. If still failing, continue alternate increases in the two threshold values.

Figure 5: Simple Dynamic Threshold Strategy

smaller than α and that the 2-norm condition number of each block (which is the ratio of σ_1/σ_s), will not be more than ρ^{-1}.

The results in the following Section 6 are based on using BILU(k) and ILUT, as described in this section, along with non-restarted GMRES. Although this is work in progress, it nevertheless illustrates great strides in our ability to solve realistic, challenging problems.

5.4.3 Strategies for Global Reordering

Poorly-ordered equation systems can inflate poor conditioning. Commensurately we have found that the reverse Cuthill-McKee (RCM) reordering [27] is an inexpensive means to improve the effectiveness of incomplete factorization preconditioners for many problems. As its name implies, it relies on the Cuthill-McKee (CM) algorithm [17] by first computing the CM ordering and then reversing the order.

Many matrix reorderings consider the sparse matrix as a graph of vertices and edges. Assuming a matrix has symmetric structure (which can be accounted for if not symmetric), a matrix graph is defined by the equations in the matrix, such that there is a vertex for each equation, and an edge exists between vertex i and vertex j if there is a nonzero matrix entry a_{ij}.

Given this graph interpretation of a matrix, starting with an arbitrarily chosen vertex, CM uses a breadth-first search to identify level sets in the matrix graph

such that each level set is the collection of as yet undeclared graph vertices that are neighbors (share an edge with) vertices in the previous level set. Ordering by level set in this fashion has a tendency to move nonzero off-diagonal matrix entries near the main diagonal. RCM is an improvement over CM in that the reverse ordering tends to reduce required fill-in for an exact factorization and therefore tends to minimize the discrepancy between an incomplete factorization and the exact factorization.

We use RCM as a means of improving the initial matrix system as needed. As we demonstrate below, the condition number in some applications is greatly reduced while in others RCM can lead to condition number inflation. Reasons for this are discussed where appropriate.

6 Test Problems

Perhaps the largest barrier to routine usage of iterative solvers on indefinite matrix systems is the wide variability in solver/preconditioner parameters required from problem to problem; this variability arises from a variety of sources which we now are just beginning to understand. As discussed in Section 3 and in Section 4, poor matrix attributes arising from scaling on a block level, variability of scaling across blocks, indefiniteness caused by zero-diagonals and diagonally-weak constraints, and mesh/matrix structure and equation ordering, etc., can complicate solver selection. The examples here demonstrate that in order to successfully model applications, many detailed choices in the solver specifications are required for no apparent reason. These choices include whether to reorder, whether and how to scale locally/globally, how much fill-in of ILU factors is required and how much fill-in is too much, whether to control partial fill with a tolerance or a band level, whether to pursue block-level preconditioning versus point or global level, how large of block is appropriate, what size of subspace is best, etc. Unfortunately it is not a simple matter of retaining enough ILU-fill to solve the problem, since the higher fill levels add rapidly to memory cost and can further undermine the condition of the preconditioning matrix. Moreover, these choices cannot be made independently of the nonlinear solution strategy, as issues of exactness at the beginning of that process effect the convergence at latter stages (cf. Section 6.3).

Our first two examples focus on the basic Navier-Stokes residual equations using the mixed $v - p$ Galerkin finite element method (GFEM). These examples are the classic four-to-one contraction and lid-driven-cavity flow problems. Solutions to the four-to-one contraction problem illustrate some of the dubious aspects of pressure stabilization schemes in confined flows; we find that improving those schemes with adjustments to the stabilization parameter lead to undermined iterative solver convergence, thereby strengthening the need to pursue strategies that employ an LBB element approach. Solutions to the lid-driven cavity problem are pursued over a range of Reynolds number to establish a benchmark for successful and robust application of iterative solvers. We achieve this benchmark by deploying a direct solver. The third example complicates matters with an incompressible solid material and a moving mesh solved together with the Navier-Stokes system. These features lead to

a variable scaling problem which requires specific solver choices. Our fourth example will examine a realistic application of incompressible flow in a flexible die cavity. We will demonstrate with this problem again why the highly touted PSPG scheme discussed in Section 4.4 is not the best choice if the matrix-conditioning improvements that PSPG provides were not the main goal. In our fifth problem we take on a large-scale three-dimensional capillary free-surface problem which illustrates some of the issues we encounter with poorly-conditioned distinguishing conditions. We finish with a viscoelastic flow problem which produces the most challenging matrix system we have solved to date. There we exemplify how adaptable block size and SVD are used to tune the preconditioners to meet the challenge, but not without a heavy cost in memory and compute time.

In all cases we employ Krylov subspace size for GMRES (c.f. Section 5) of 100 to 500. We find this "knob" can also have a significant effect, as expected, beyond our choice of preconditioners. However, our main focus remains the preconditioners, and our ultimate goal to reduce and eventually eliminate guesswork. As previously discussed we limit our experiments to the ILUT(x) (x indicating a factor of nonzero fill values beyond the basic storage) and BILU(k) preconditioner classes. On top of these we will employ preconditioner matrix perturbations following the strategy outline in Figure 4 and 5, global matrix reordering with RCM, and adaptive block size control where necessary to achieve acceptable performance. These tactics are used over and above the increase-level-of-fill tactic as expedients to save memory cost. We follow the same sequence of solves for each problem so that we can compare results at the end. We should point out that it is advantageous to adjust the residual ratio tolerance, which is the ratio of the L2 norm of current matrix residual $Ax - b$ to the starting matrix residual $Ax_0 - b_0$, especially for early Newton iterations.

6.1 Navier-Stokes System with Inflow and Outflow Boundary Conditions

Our first test problem is designed to illustrate the accuracy of the LBB-compliant mixed interpolation velocity pressure formulation and the solution degeneracy that can result from improper usage of pressure-stabilized schemes (as discussed in Section 4.4). Our first problem addresses incompressible flow in a three dimensional four-to-one contraction geometry. The geometry and computational mesh is shown Figure 6 and results in 37,000 unknowns. Figure 7 shows the computed channel-directed velocity component along the symmetry plane. The three-dimensional case was contrived from the base two-dimensional planar case by simply extruding the mesh in the z-direction, as indicated in the figure. We use symmetry boundary conditions on the front plane, and no-slip/impenetrability conditions on the back plane. In this way we recover the two-dimensional solution on the front symmetry plane but provide a truly three-dimensional test as a better measure of the compute-resource requirements. Here we utilize the baseline case Q1P0 element type (an LBB element) without pressure stabilization as a benchmark. We compare the baseline case with the Q1Q1 stabilized velocity-pressure approach, for different constant multipliers on

the least-squares term (cf. α in Eq. 29). As a second benchmark we show the same velocity profile in the two-dimensional test case as computed by the commercial code FIDAP [39], which has implemented the same stabilization term.

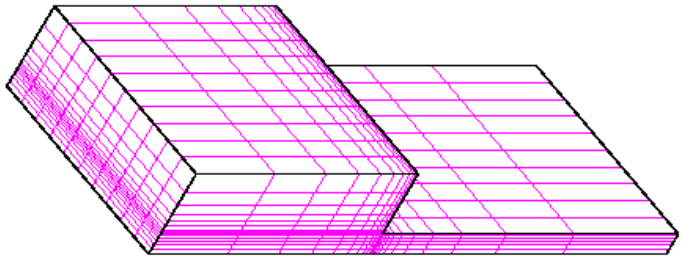

Figure 6: Mesh for three-dimensional 4:1 contraction flow with bottom and side symmetry.

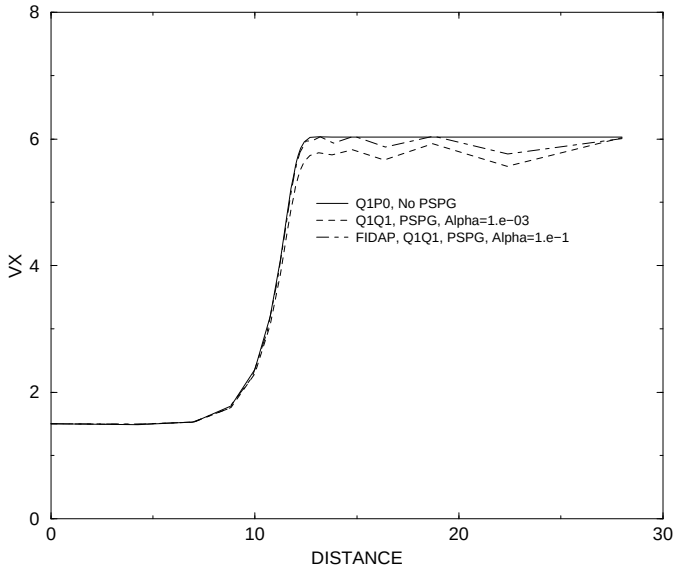

Figure 7: X-Velocity component along centerline symmetry plane for 4:1 contraction flow, Re = 10.

In both cases FIDAP and our in-house incompressible flow code have been verified on this simple test problem, using the LBB compliant Q1P0 element type. Notice in Figure 7 that when $\alpha < 10^{-3}$ the Q1Q1 element type with pressure stabilization yields a fairly accurate solution. This is not the case when α is as high as 0.1. The x-component of the velocity along the centerline shows evidence of mass sources and sinks, viz. wiggles, in this case. This highly unacceptable solution clearly demonstrates the need for restrictions on our choice of the pressure stabilization parameter α. Because the focus of this paper is solver efficiency and robustness, we now examine how the choice of α affects iterative solver performance. The difference in profiles between the FIDAP solution and our code are likely due to

	Preconditioners		
Element	ilut(1.0)	ilut(**2.0**)	bilu(1), Threshold = 10^{-9}
Q1P0 LBB	no convergence	no convergence	**CN**=1.0e+20 CPU time 300 s 358 its
Q1Q1 PSPG $\alpha = 0.1$	**CN**=1.0e+2 CPU time 190 s 155 its	**CN**=1.0e+2 CPU time 320 s 50 its	**CN**=6.0e+2 CPU time 200 s 173 its
Q1Q1 PSPG $\alpha = 10^{-3}$	no convergence	**CN**=1.0e+4 CPU time 320 s 55 its	**CN**=6.0e+3 CPU time 300 s 45 its
Q1Q1 PSPG $\alpha = 10^{-5}$	no convergence	no convergence **CN**=1.0e+16	no convergence **CN**=1.0e+21

CN - Condition Number of Preconditioner; **k** - Fill level for bilu;
x - Fill level for ilut; **its** -Average Number of Matrix-solver Iterations.

Table 1: Convergence results for four-to-one contraction flow.

the way in which the stabilization parameter τ is applied. We use a global average element size and keep this parameter constant whereas FIDAP uses a local measure. In either case, however, the same degradation in the solution quality is evident at the same value of α.

Table 1 gives the memory requirements, CPU times, and in some cases the condition number of the preconditioning factors for three variants of the incomplete factorization ILU methods (cf. Section 5 for a complete discussion of ILU-preconditioning). We reserve most of our discussion on performance and robustness of preconditioning strategies to the remaining test problems, but give some preliminary results here to set the stage for that discussion. ILUT(0) preconditioning basically fails in all cases except that which has a high pressure stabilization factor, viz. $\alpha = 0.1$. Clearly we will not be able to rely on this preconditioner because it only works when the solution continuity constraint is not satisfied. ILUT with a level-of-fill of 2.0 gives more reliability to the solver, enabling solutions for α as low as 10^{-3}, but is still inadequate for the general case as the solution quality is less than that attainable with the LBB element, at the same level of mesh refinement. It should be noted that the memory requirements for two levels of fill in ILUT is enormous for a small three-dimensional problem. The condition number for the LBB element case of the preconditioner is greater than 10^{20}, thus large perturbations to the diagonal are required for BILU. Notice that in all but the $\alpha = 10^{-5}$ case we were able to obtain a solution with BILU preconditioning. It was quite revealing however that the condition number depends highly on α.

The pressure stabilization term becomes more accurate for a Q1 velocity element at higher Reynolds number (viz. the more important advective momentum transport) and higher levels of mesh refinement. The three-dimensional case happens to correspond to a Reynolds number of 10. Unfortunately, we still see a large degradation in iterative solver performance for smaller values of α. This degradation can be prohibitive, as we later demonstrate in Section 6.3. In many of our applications the high values of α required for convergence generate errors that render the solutions

unacceptable.

6.2 Navier-Stokes System with Dirichlet Boundary Conditions

Perhaps one of the most often-used test cases for incompressible, low Reynolds number flow is that of a finite rectangular cavity with an infinite planar lid sliding along one boundary. The geometry, boundary conditions and finite element mesh are pictured in Figure 9. Although the technological relevance of this exact flow geometry is debatable, it serves as a worthwhile numerical test because it exhibits useful features and challenges. First, it is a problem that involves only Dirichlet boundary data and thereby is not confounded by the placement effects of inflow and outflow boundaries. Second, sharp corners at which boundary data is not continuous lead to singularities in the stress; singularities lead to steep gradients in the solution and non-smooth flow, thereby making it more relevant to a wide variety of practical flows. Third, the pattern of fluid flow exhibits structure changes and flow features that challenge many numerical techniques, e.g. recirculation in the corners and bifurcations in flow structure with increasing Reynolds number. Finally, because of the absence of inflow and outflow boundaries, there is no natural pressure datum, which forces the practitioner to specify such a datum somewhere in the cavity.

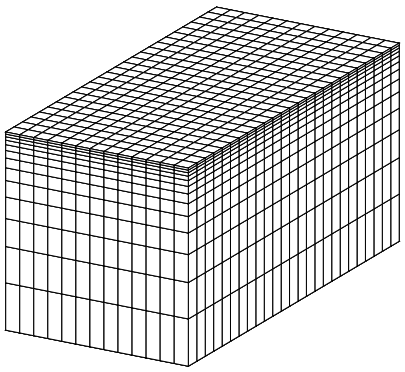

Figure 8: Lid-driven cavity computational mesh

We consider here a box of width L, height L, and length 2L and take the dimension L to be unity. Our base mesh contains 13 elements along the width and height, and 25 elements along the length. The mesh spacing is reduced continuously towards the lid to better resolve the non-smooth solution behavior in the corners. We take the viscosity μ and the density ρ as unity. We vary the lid speed U from 10^{-5} to 1000. It is appropriate to define a Reynolds number as $\mathrm{Re} = \frac{U\rho L}{\mu}$. In the

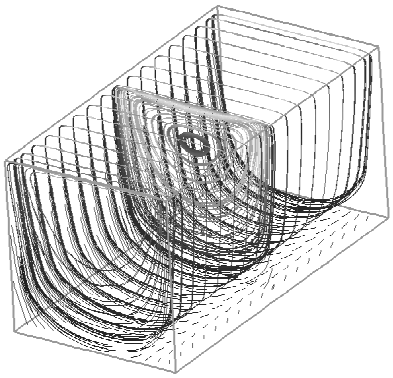

Figure 9: Pattern of flow for three-dimensional lid-driven cavity problem. $Re = 10$.

results and discussion to follow we will refer solutions parameterized by Reynolds number, which is equivalent to the lid speed.

We have chosen to keep the Krylov subspace size at 200. Previous tests show that sizes of around 10 to 20 led to severe degradation in convergence, and sizes much greater than 200 seem to offer little benefit in terms of linear solver convergence rate, but this depends on the mesh size. Our convergence tolerance for the nonlinear Newton iteration loop is taken as 10^{-9}, and is measured as the L2 norm of the Galerkin weighted residual vector. As for the iterative solver, we measure the extent of convergence with a residual ratio, which is the ratio of the L2 norm of current matrix residual $Ax - b$ to the starting matrix residual $Ax_0 - b_0$. In this test problem we set the threshold value to be 10^{-3}.

Here as in all the examples we examine the performance and computational requirements of the following preconditioners: ILUT(1.0), ILUT(x), and ILUT(x) with perturbations, BILU(0), BILU(k), and BILU(k) with perturbations. For ILUT we choose x based on the minimum level required for successful solution; likewise for BILU, we choose n based on the minimum level required for successful solution. In some cases, however, we exceeded any practical value of n or x due to increased memory cost, as noted. The perturbation thresholding strategy (cf. Figure 4 for ILUT and Figure 5 for BILU) involves two additional parameters we call absolute and relative thresholds. In this first study we will perform each of these tests with and without RCM to show the importance of reordering.

We show results at Re = 10 in Table 2 for Q1P0 and Q2Q1 LBB compliant velocity-pressure elements. Problem size details are given in the Table. Note that in the Q2Q1 case the mesh results in a much higher problem resolution, as we keep the number of elements the same. In all cases we solved the steady equations with a zero initial guess. Reported in this and all following tables as indicated are the condition number of the preconditioner matrix (if successfully computed), the number-of-nonzeros stored, the average number of iterations per Newton step for the iterative solver solution, and the total matrix solution time. Surprisingly and somewhat fortuitously we found that a solution was possible with a level-of-fill of 1.0, with ILUT. In all cases the level-of-fill increase only added to the memory cost with little or no improvement in performance; this behavior is to be expected in

	Preconditioners					
Element and DOF	ilut(1.0)	ilut(3)	ilut(1) with perturbations	bilu(0)	bilu(1)	bilu(1) with perturbations
Q1P0 (no RCM) 19513 DOF	unconverged **CN** = 8.9e10 **nnz**=1325137	unconverged **CN**=3.7e16 **nnz**=3791511	**CN**=8.5e03 **nnz**=1325137 100 **its**/54s	unconverged **CN** = 5.3e11	**CN** = 1.5e04 **nnz**=1325137 30 **its**/100s	**CN** = 2.1e04 **nnz**=1325137 16 **its**/68s
Q1P0 (RCM) 19513 DOF	zero pivot	zero pivot	**CN**=4.3e22 **nnz**=1325137 200 **its**/220s	unconverged **CN** = 5.3e11	**CN** = 3.0e4 **nnz**=1325137 30 **its**/25s	**CN** = 2.0e4 **nnz**= 1325137 20 **its**/68s
Q2Q1 (RCM) 112215 DOF	**CN** = 1.e8 **nnz**=23572369 85 **its**/4000s	not enough memory	**CN**=1e5 **nnz**= 23572369 31 **its**/3000s	pivot failed	pivot failed	weak converge **CN**=1e5 **nnz**=23572369 200+ **its**/5000s

CN - Condition Number of Preconditioner; **k** - Fill level for bilu;
x - Fill level for ilut; **its** -Average Number of Matrix-solver Iterations.
nnz - Number of Nonzeros Stored (Preconditioner)

Table 2: Convergence results for Lid-Driven Cavity problem(∞).

light of the relatively small condition number. Some interesting behavior regarding global reordering is noteworthy. With RCM reordering, Q1P0 elements resulted in zero-pivots at all levels of fill; fortunately acceptable behavior could be regained by using the threshold parameters. The opposite behavior was observed for Q2Q1 velocity-pressure elements: RCM was required to avoid pivot problems in the ILUT scheme. Contrary to the Q1P0 case, thresholding *without* RCM allowed for acceptable convergence.

To measure the iterative solver performance we compare all cases to that of a direct solver, in the Q1P0 case. By "performance" we imply robustness and not compute time or memory cost, as that comparison is already understood. We can state that the results of performing first-order continuation with Q1P0 elements starting from Re = 1 and going in increments of 300 to 1200 as follows. With a direct solver we were able to continue in steps of 300 up to 1200. With the iterative solver choice that seem to perform best according to Table 2, viz. ILUT in row 1 together with a residual ratio tolerance of 10^{-7}, we observed erratic convergence past Re = 600 and never achieved Re = 1200 without reducing the step size to an impractically low level. This behavior indicates not only that iterative solvers are fickle, but that solution scheme exhibits a lack of robustness. The reason for this can be explained with the matrix condition number estimate: the matrix condition degrades the further from the solution (viz. the larger the parameter step size) and the poorer the initial guess, often dooming the iterative solver regardless of the preconditioner.

These numerical tests on the ubiquitous lid-driven cavity problem illustrate several key issues with respect to our ultimate goal of achieving the robustness of a direct solver. For one, we find that fill-reducing orderings like that provided with RCM are essential in the Q2Q1 case, but greatly inflate the condition number in

the Q1P0 case. We suspect the reason is that the matrix graph for the Q1P0 case has a disconnected entry at the node center where only pressure degrees of freedom exist(cf. Figure 1), and so reordering proceeds without seeking to keep these matrix entries near the diagonal. Fortunately we can repair the detrimental effects of RCM with matrix perturbation thresholds, as shown in row 2 of Table 2. Notice that the condition number of the preconditioner is the indicator, as its value in failed cases is extreme, sometimes exceeding 10^{15}. In those cases, thresholding can be used to achieve a nearly-manageable matrix system for ILUT or BILU. Finally, when compared side-by-side with a direct solver in terms of robustness, we find that the matrix system conditioning is affected by the size of the parameter step, indicating that larger step sizes require more solver accuracy to achieve the same performance. In this example we were unable to achieve this accuracy with a reasonable amount of computational work and memory using iterative solvers. This reconfirms that oftentimes increasing the level of fill is not a viable approach since memory limitations are rapidly exceeded. This defeats one of the main reasons for deploying iterative solvers to begin with; their low memory usage.

6.3 Navier-Stokes System with coupled Fluid-Structure Interaction

The two examples in this section feature fluid-structural interaction (FSI) phenomena. The roll metering nip problem we cover first exemplifies some difficulties that may arise in practical applications, even though it is relatively small and two dimensional. The die-flexure problem covered at the end of this section provides a good test for a typical design problem of substantial size.

In this particular roll-nip metering flow problem liquid is entrained between two deformable co-rotating rollers (cf. Figure 10). The high metering pressures in the liquid due to rapidly converging flow can lead to deformation of the incompressible rubber solid. Our calculations employ a Q2P0 velocity-pressure element in the liquid ALE region and a Q2P0 displacement-pressure interpolation in the Lagrangian solid regions. Material properties in the solid correspond to a standard rubber-covered roll hardness; the liquid is taken to be Newtonian with a viscosity of 6 P. The outer diameter of both rolls is 0.14 m and the linear roll speed of the surfaces is 1 m/s in the stress-free state. In total this two-dimensional problem consists of about 6000 unknowns.

Convergence results are shown in Table 3. The first row corresponds to an unordered matrix system. Note that the condition number is moderate (order 10^{10}) across the row and hence, as expected, the threshold perturbations to the preconditioner do not affect convergence significantly. Quite surprisingly though, ILUT and BILU preconditioning required a level of fill around four for successful problem solution. In fact we were completely unsuccessful for all matrix-graph fillin levels below 7.0 for ILUT and at all levels of fill for BILU, using the standard solver settings specified in the beginning of Section 6; luckily we were able to achieve solutions at lower levels of fill by requiring more stringent matrix system convergence on the

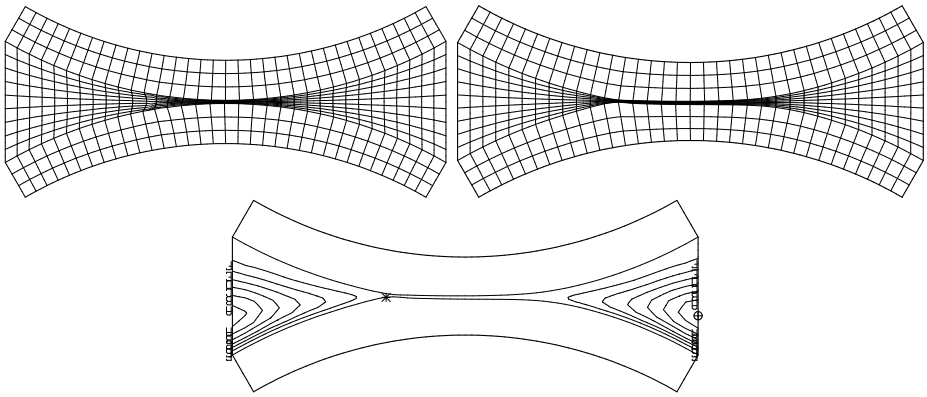

Figure 10: Deformable roll-nip metering flow. Undeformed mesh (top); deformed mesh (middle); pattern of streamlines (bottom)

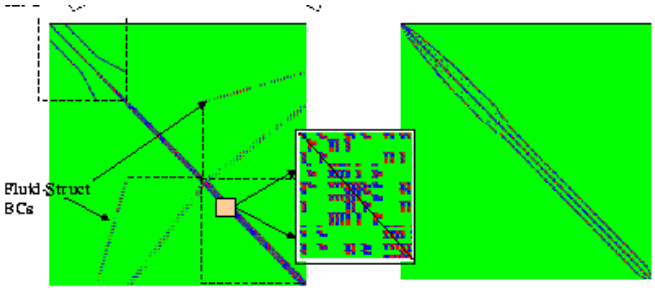

Figure 11: Matrix structure for deformable roll-nip problem

first Newton iteration, viz. reducing the residual ratio tolerance to 10^{-15} enabled solutions with ILUT and BILU to be obtained with $\mathbf{x} = \mathbf{4}$ and $k = 4$, respectively.

Reordering the resulting matrix with RCM, in this case, undermines convergence ostensibly due to the destruction of the naturally material-based ordering arising from assembly. The result of reordering is also depicted in matrix graphs in Figure 11. The graphs show the existence of zero-diagonals resulting from the incompressibility constraints. In the unordered case one can pick up the structural effects of assembly based on material blocks, since assembly is done block-by-block. The fluid-structural boundary conditions lead to "wings" in the unordered structure as they involve velocity components in equation regions that have strictly displacement-based elastic stress balances. Although RCM reordering clearly reduces the bandwidth of the problem, we find that global matrix conditioning is worsened (cf. Table 3). In the reordered case we were completely unsuccessful with ILUT and had to rely on BILU for solution. Noteworthy also is the lack of benefit realized from matrix preconditioner perturbations, even with the high initial preconditioner condition number.

Overall the results for this problem are somewhat disappointing as such a high fill level is usually impractical in three dimensions due to a high memory cost, unless parallel processing is used to reduce this cost without undermining the effectiveness of the preconditioner. We speculate that some adaptive schemes that weigh level-of-

CN - Condition Number of Preconditioner; **nnz** - Number of Nonzeros Stored (Preconditioner); **k** - Fill Level for BILU; **x** - Fill Level for ILUT; **its** -Average Number of Matrix-solver Iterations.						
Preconditioner	ilut(1.0)	ilut(**x**)	ilut(**n**) with perturbations	bilu(0)	bilu(**k**)	bilu(**k**) with perturbations
no RCM 6000 unknowns	**CN** = 3.e+10 **nnz**=418343 no convergence	**n**=4 **CN**=5.1e+10 **nnz**=1547849 70 **its**/100 s	weakened convergence	**CN** = 3.e+16 **nnz**=585680 no convergence	**n**=6 **CN**=4.e+10 **nnz**=1673532 7 **its**/10s	**n**=4 A/R = 1.e-09 **CN**=2.e+10 **nnz**=1338697 300+ **its**/100s
RCM 6000 unknowns	zero pivot	zero pivot	no convergence fails	no convergence	**n**=4 **CN**=2.e+13 **nnz**=1338697 120 **its**/38 s	**n**=4 A/R = 1.e-09 **CN**=2.e+10 **nnz**=1338697 300+ **its**/100s weak convergence

Table 3: Convergence results for Deformable Roller Nip Problem.

fill, material block reordering, and residual ratio tolerance will be required for more robust solution. Incidently, direct solvers have no trouble with this matrix system. Perhaps another effective approach would be to reorder the equations based on material block, which will be considered in future work.

This erratic and somewhat disappointing performance of ILUT and BILU indicates the challenges we must surmount to obtain performance and reliability equivalent to that of a direct solver. Luckily, the condition of the matrix systems from large-deformation fluid-structure interaction problems with incompressible fluid and solid materials has been one of the most formidable to date. In the next example we simplify matters with regard to the physics by considering linear elasticity in the solid region, no incompressibility constraints in the solid, and small deformations.

The die-flexure problem pictured in Figure 12 is of prime importance to the polymer processing and continuous liquid thin film coating industries. The goal is to extrude thin liquid films or sheets with as uniform a thickness as possible in the cross-die direction. The solid regions depicted in the figure make up what is commonly called a slot-die. These dies are usually center-fed and consist of a long and generally tapered chamber plenum attached to which is a thin narrow slot, both running the length of the die. The plenum is tapered in size to keep the pressure drop across the slot as uniform as possible. Determining the taper geometry, slot length and other geometric parameters for a variety of liquid rheologies and flow conditions that result in a uniform film is often the major objective of die designers. One additional complication highlighted by this example is the possibility that high liquid pressures may result in the die bowing out across the long span, which can greatly undermine film uniformity.

Our calculations pertain to the die-slot geometry pictured in Figure 12. We restrict our calculations to purely-viscous Newtonian or shear-thinning liquids. The results shown here are for a Reynolds number Re = 1, based on the speed of the liquid at the inflow and the width of the plenum chamber. We found no additional complications with respect to the iterative solvers resulted from shear thinning or

moderately higher Reynolds number flows (up to 50). The meshes employed lead to 77000 unknowns for the Q1P0 mesh, and approximately 79000 for the Q1Q1 stabilized case. As with the four-to-one contraction example above, we show results for a Q1Q1 PSPG case at three different values of α (cf. Section 4.4). By "flow-only" we mean that the calculations were done without the solid die and with no fluid-structure interactions. This case alone proves to be quite challenging. We then compare the PSPG results with the same flow-only case but with a LBB-compliant Q1P0 velocity-pressure element. Finally we perform the complete calculation, including the die-flexure, as the ultimate test. In that case, we are hard pressed to find a suitable preconditioner.

The first row of Table 4 corresponds to the flow-only Q1P0 case. ILUT(1.0) preconditioner condition number has a value of order 10^{16} and ILUT(2.0) of order 10^{43}. In both cases the calculation encountered zero pivots on subsequent iterations and no solution was obtained. Increasing the fill beyond 2.0 offered no benefit, at least without matrix perturbations. As in the previous two examples, reordering with RCM increased the preconditioner condition number to a more unmanageable value. Strong convergence was obtained by lowering the condition number with our perturbation strategy, and increasing the fill level to 2.0. The absolute and relative perturbations did not improve convergence at lower fill levels. With five Newton iterations and with an average of 162 linear solve iterations per Newton step we obtained a solution in about 4100 s CPU on a Sun UltraSparc workstation. Interestingly, BILU also resulted in good convergence, even with $n = 1$, albeit taking a substantially longer time due to a higher iteration count. Perturbations with BILU seemed to undermine convergence and in fact convergence was never strongly obtained. The performance of ILUT(2.0) was weakened with higher absolute and relative perturbations alone. Here the proper combination of small perturbations and fill provides the optimum selection, i.e., using perturbations to lower the condition number and increasing the level of fill to improve convergence.

In row 5 of Table 4 we again employ the Q1P0 element but include the entire fluid-structural interaction problem. The number of unknowns in this case was approximately 80,000. Memory cost is a substantial issue when pursuing problems of this size, and in fact a problem any larger forces us to parallel computations, the subject of future work. Clearly to keep the unknown count under 100,000 we had to reduce the mesh density for this calculation. In this case we find extreme preconditioner condition numbers for ILUT(1.0), of order 10^{26} and eventually failed pivot strategies at all levels of fill. We had some success in lowering the condition number with matrix perturbations, achieving a solution at a fill level of 3.0 with an absolute perturbation of 10^5 and a relative perturbation of 1.01. In a similar fashion as with the deformable roller nip problem, we had to lower the residual ratio tolerance to 10^{-15} on the first iteration to achieve a solution. Fortunately, the matrix system from this complete problem was also solvable with BILU(2), together with an absolute perturbation and relative perturbations of the order of 10^{-5}. However, in this case the convergence was slow, with the iterative convergence stagnating close to the solution. Moreover, nearly 8000s of matrix solve time was required over five Newton steps, clearly indicating the need for parallel computing.

CN - Condition Number of Preconditioner; **nnz** - Number of Nonzeros Stored (Preconditioner); **n** - Fill Level; **its** -Average Number of Matrix-solver Iterations; **A/R** -Absolute and relative thresholds applied to preconditioner.						
Preconditioner	ilut(1.0)	ilut(**x**)	ilut(**x**) with perturbations	bilu(0)	bilu(**k**)	bilu(**k**) with perturbations
Q1P0 77,000 unknowns	**CN** = 3.e+16 **nnz**=1947453 zero pivot	**n**=2 nnz=3836482 **CN**=1.e+43 zero pivot	**n**=2 nnz=3836482 A/R=10^{-5}/1.0 CN=4.e+10 162**its**/800 s	**CN** = 3.e+5 **nnz**=5647613 300+ **its**/6200s	**n**=2 **CN**=3.4e+4 **nnz**=16129034 175 **its**/6000s	**n**=1 A/R=10^{-5}/1.0 **CN**=1e+4 **nnz**=5647613 300+ **its**/1000+ s weak convg
Q1Q1 79,000 unknowns $\alpha = 10^{-3}$	**CN** = 6.e+5 **nnz**=2871184 40 **its**/350s	**n**=2 **CN**=4.e+6 **nnz**= 5684944 30 **its**/1000s	**n**=1 A/R=10^{-5}/1.0 CN=1.e+03 97**its**/630 s	**CN** = 1.e+5 **nnz**=6060093 71 **its**/600s	**n**=2 **CN**=6e+5 **nnz**=10965882 65 **its**/850s	**n**=1 A/R=10^{-5}/1.0 **CN**=2e+1 **nnz**=5647613 all fails
Q1Q1 79,000 unknowns $\alpha = 10^{-4}$	**CN** = 10^{23} no convergence	**n**=3 **CN**=1.e+8 **nnz**=7752196 72 **its**/2200s	**n**=3 A/R=10^{-5}/1.0 CN=1.e+05 83**its**/2500 s	**CN** = 10^{23} pivot fails	**n**=3 **CN**=10^{23} fails	**n**=4 A/R=10^{-5}/1.0 **CN**=3.e2 all fails
Q1Q1 79,000 unknowns $\alpha = 10^{-5}$	**CN** = 10^{33} no convergence	**n**=2 zero pivot	**n**=3 A/R=10^{-3}/1.0 CN=1.e+03 **nnz**=7752196 200+**its**/5500 s	**CN** = 10^{32} pivot fails	**n**=2 pivot fails	no convergence on all attempts
Q1P0, Whole FSI problem 80,000 unknowns	**CN** = 10^{26} zero pivot	zero pivot	x=3.0 nnz=22481368 A/R=10^{-5}/1.01 CN=8.1e+05 190 **its**/2400s	**CN** = 10^{7} zero pivot	zero pivot no convergence	**n**=2 A/R=$10^{-3}/10^{-3}$ **CN**=6.e4 120 **its**/7800s

Table 4: Convergence results for die-flexure problem.

Finally, in rows two, three, and four we include the results for PSPG Q1Q1 elements. Here the performance of the iterative solvers is exceptionally good for the $\alpha = 10^{-3}$ case, with solution time much smaller than for any Q1P0 case. However, as α is reduced to 10^{-5}, the preconditioner condition number rises precipitously, reaching 10^{33} at $\alpha = 10^{-5}$. In fact, we obtained matrix solutions in this last case only for perturbations of ILUT(3.0). At this fill level the number-of-nonzeros stored led to excessive memory requirements. Perhaps the most disconcerting aspect of PSPG approaches for this problem, however, come from the quality of the solution.

Figure 14 shows the profile of x-directed velocity component in the plenum and slot regions on a cut along the bottom symmetry plane at mid-die, as depicted in Figure 13, for the Q1Q1 element with PSPG at three values of α (cf. Eq. 29). For comparison we also include results for the LBB-compliant Q1P0 element. It is noteworthy that in order to achieve the accuracy of a LBB element, α needs to be less than about 10^{-5}. Unfortunately, as we observed in Section 4.4 the iterative solvers perform poorly at these values. Note that the outflow velocity is the same in all cases, but the pressure-stabilized elements are plagued by wiggles in the slot where the pressure gradient is large. The detrimental effect of PSPG in this problem can be fixed with mesh refinement or by evaluating the higher-order terms of Equation 29, both which come with substantial computational costs.

6.4 Capillary Free Surface Flows

The example presented in this section has proven to be a considerable challenge for methods discussed so far. It typifies some of the problems that boundary conditions can introduce into matrix system.

Figure 15 depicts the flow geomety under consideration. A Newtonian fluid wells up in a thick-walled trough, eventually overflowing the sides. Liquid films drain down both exterior sides of the trough under gravity, eventually detaching to fall freely as liquid sheets. Predicting the position and motion of the capillary free surface is challenging; the problem is very nonlinear and geometrically complex, as the surface moves and curves in three dimensions. A very general finite element method based on the work of Sackinger et al. [53] (cf. Section 3.1) and extended to three dimensions by Cairncross, et al. [10], is employed for this purpose. Details of the technique, togther with other three dimensional applications can be found in these references.

By itself, the moving mesh problem amounts to little more than a Laplace problem for the mesh node displacements. However, coupling the shape of the free surface to the motion of the mesh has a very detrimental effect on the condition of the matrix system. The so-called kinematic constraint for a free surface, viz. Eq. 13, enforces the requirement that the free surface be also a material surface of the flow. However, in this mesh deformation method this constraint appears as a force on the "pseudo-solid" mesh material, not a constraint on the fluid momentum equation. Unfortunately, the kinematic constraint is only weakly dependent on the displacements of the surface nodes and can introduce poor-conditioning into the matrix system.

All computations presented here use eight-noded hexahedron meshes. A stabilized Q1Q1 velocity/pressure formulation with $alpha = 0.5$ was used for all computations. This large value of the stabilization weight proved necessary for convergence in all cases. In contrast, to cases previously presented (cf. four-to-one-contraction flow and die-cavity flow), however, a large value could be used without introducing significant mass errors. The free surfaces minimize pressure gradients and the viscous forces are significantly larger than gravitational. Inertial forces were also insignificant. As a consequence, the Laplacian of the velocity field was generally small at most points within the flow field and as a result the mass-error introduced by the stabilizing term was not great.

Attempts to use LBB-compliant elements were problematic. The large number of unknowns and high-bandwidth associated with the Taylor-Hood Q2Q1 mixed interpolation elements tax both available computational resources and the patience of the analyst. Parallel processing would have alleviated these to some degree at the expense of adding scaling issues. In the end, it was decided to not consider this element. The Q1P0 element does not present the same issues. However, as in all of our case studies, this element results in very poorly-conditioned matrices–so much so that iterative solution of the matrices to the same degree of accuracy as the Q1Q1 element was not possible. It was decided to present results only for the stabilized Q1Q1 element.

	Preconditioners					
Element and DOF	ilut(1.0)	ilut(x)	ilut(n) with perturbations	bilu(0)	bilu(x)	bilu(x) with perturbations
Q1Q1 (no RCM) 15,235 DOF	**CN** = 1.4e+05 **nnz**=9642808 490 **its**/1800s	zero pivot	**x** = 1.0 **CN**=3.9e+07 **nnz**=9642808 **A/R**=1.e-5/1.01 479 **its**/1744s	**CN** = 1.6e+07 **nnz**=23264090 500 **its**/2500s	**n** =1 **CN** = 1.3e03 **nnz**= 41811896 500 **its**/3600s	**n**=0 **A/R**=1.e-1/1.0e-1 **CN** = 6.9e02 **nnz**=23264090 464 **its**/2200s
Q1Q1 (RCM) 15,235 DOF	**CN** = 1.1e+09 **nnz**=9642808 460 **its**/1719s	zero pivot	**x** = 1.0 **CN**=8.5e+08 **nnz**=9642808 **A/R**=1.e-5/1.01 336 **its**/1180s	**CN** = 1.6e+07 **nnz**=23264090 500 **its**/2500s	**n** =1 **CN** = 1.3e03 **nnz**= 41811896 500 **its**/3600s	**n**=0 **A/R**=1.e-1/1.0e-1 **CN** = 6.9e02 **nnz**=23264090 464 **its**/2200s

CN - Condition Number of Preconditioner; **k** - Fill level for bilu;
x - Fill level for ilut; **its** -Average Number of Matrix-solver Iterations.
nnz - Number of Nonzeros Stored (Preconditioner)

Table 5: Convergence results for Lid-Driven Cavity problem(∞).

The illustration in Figure 15 represents a convergent result from the non-linear solver. This was chosen as a testbed for the preconditioners by resolving the problem with the identical material parameters but changing solver parameters. Convergence of the non-linear problem thus occurred in a single iteration and the iterative solvers were always presented with the same basic matrix structure. In all cases shown a Krylov subspace size of 500 was used with modified Gram-Schmidt orthogonalization. Scaling of the matrix was done by absolute row sums. No more than 500 iterations were permitted to achieve a residual ratio tolerance of 10^{-6}.

A summary of the results for the ILUT preconditioner is presented in Table 5. Interestingly, without perturbations to the matrix system, only a fill-level of one is effective. For higher levels-of-fill, a zero pivot is the consistent result. This is true regardless of whether RCM reordering is employed. However, it is perhaps misleading to suggest that even the lowest level of fill is effective, since the number of iterations required with ILUT(1) is very nearly at the upper limit of 500, and the time taken is on the order of one-half hour. As before, zero pivots or aborts of the solver always occur for levels-of-fill greater than one (with one exception discussed below); although, the presence of perturbations does improve the performance of the iterative solvers with respect to the same case without perturbations.

One exception to this result is noteworthy. For ILUT(4.0) with RCM reordering and perturbation parameters as noted in Table 5 there is a dramatic decrease in the condition number to approximately 10^4 and a corresponding drop in the number of iterations to 45. Unfortunately, the time to solution is only slightly better and the memory requirements were roughly four times as severe, making it dubious as to whether this represents an improvement. This result also points to the unpredictable nature of these solution procedures. One cannot deduce for the behavior at one level-of-fill what will happen at a different level-of-fill.

A summary of the results using the BILU preconditioner appear in Table 5. Unlike the ILUT preconditioner, BILU does not halt for fill levels larger that one.

However, the convergence rates are slow. Without perturbations, convergence was not achieved in the requisite 500 iterations, irrespective of the level of fill value. Inclusion of perturbations permitted convergence within 500 iterations. However, this occurred only using the relatively large value of 0.1 for the absolute and relative threshold values; smaller threshold values gave ill conditioned matrices or lack of convergence. For the convergent cases, increasing the level of fill decreased the number of iterations required, but the time to solution increased with level-of-fill as did, of course, the memory requirements.

Based upon this evidence, we can conclude that the solver techniques and preconditioners tested here are not optimal when applied to problems employing this type of mesh motion algorithm on three dimensional free surface problems. However, it is also clear that the difficulties encountered result directly from the manner in which the kinematic boundary condition is enforced. Fixed mesh problems, in which the kinematic boundary condition is applied to the fluid momentum equation, do not display such convergence difficulties. Recognizing that matrix preconditioning can occur in many places, another more effective method for enforcing the kinematic condition that would ab initio introduce better matrix conditioning would be better than developing a different preconditioner to solve these problems. Some type of stabilizing term included in the boundary condition is a likely possibility.

6.5 Viscoelastic Flow

In this example, we consider the flow of an eight-mode Giesekus fluid through a four-to-one contraction. The discontinuous Galerkin method is used to discretize the stress equation with bilinear discontinuous interpolants (cf. [26]; [3]). We use an LBB compliant element for the velocity and pressure (Q2Q1) and bilinear continuous interpolant for the velocity gradient. From experience, we have found that non-LBB compliant elements, even using pressure stabilization, are unsucessful for solving viscoelastic flow problems. This may be due to additional constraints that arise in this more complex case where the stress equation is another tensor field variable and cannot be eliminated as in the Newtonian case. From the standpoint of discretized matrix equations, an eight- mode Giesekus model implies eight additional symmetric tensor unknowns [6] for each of the four nodes in the element. In addition, we have the velocity gradient equation that is a nonsymmetric tensor. These additional tensor unknowns create an elemental stiffness matrix with a very large bandwidth compared to a Newtonian-flow problem on the same mesh. One issue that we have noticed regarding problems with discontinuous variables is that they seem considerably harder for the preconditioner/solver pairs to solve. In general, they require more memory and higher levels of fill to acheive a solution. Ideally our solver would be able to statically condense out the discontinuous variables, which would greatly reduce the bandwidth of the matrix and make the problem easier to solve.

The mesh we employed for this classic viscoelastic benchmark problem is is a refined version of the surface mesh on the symmetry plane of the three-dimensional geometry shown in Figure 6. The mesh is highly refined towards the contraction

and results in a problem of 23439 unknowns, most of which are stress variables. The material properties have been greatly simplified for this test problem: each mode has identical viscosity, time constant and mobility. The viscosity used is 1 cp, the time constant is 0.002s, and the mobility is 0.002. As is well known in the viscoelastic literature, the solution will become more difficult to acheive as the amount of elasticity of the fluid is increased, as denoted by the time constant [Guenette and Fortin, 1995]. Since our time constant is quite small, we are solving a relatively easy viscoelastic problem.

As in the previous cases we ran the problem with both the ILUT and the BILU preconditioners coupled with GMRES solver, with and without perturbations and RCM reordering. From Table 6 we can see that for this single material problem RCM is very helpful for the ILUT preconditioner. Without RCM, regardless of the level of fill, a zero pivot was encountered and the problem could not be solved. Adding the standard perturbation as described in Section 5 allowed us to solve the problem in a reasonable amount of time (315s) with relatively low memory usage (445mb). Using RCM reordering allowed us to solve the problem with the lowest level of fill for ILUT, viz. ILUT(1.0). These settings used about as much CPU time and memory as the perturbed method without RCM. Adding an additional level of fill almost doubled the number of nonzeros and memory requirements while quadrupling the solution time. Interestingly, this formulation required only two iterations per Newton step, but an distressingly large solution time, i.e. 667s per iteration. It took roughly the same solution time using a direct solver. For ILUT(1.0), the perturbation technique did not improve matters significantly.

The BILU preconditioner performed similarly to ILUT, with and without RCM reordering or perturbations. For BILU(0), the iteration count was higher than the ILUT case, but took less CPU time and more memory (894mb). Increasing the level of fill almost halved the solution time, while increasing the memory requirements to almost impossible levels (3.6gb). From Table 6 we can conclude that ILUT(1) with RCM reordering works best for this problem, giving both reasonable solution times and memory requirements. If reordering is unavailable, it may be necessary to perturbed the preconditioner matrix.

7 Summary and Conclusions

We have presented a comprehensive historical and current state-of-the-art study on the performance of iterative solvers and preconditioners on matrix systems arising from fully-coupled finite element formulations of incompressible flow. More importantly, we have made this assessment in the presence of additional complications arising in technologically important problems, e.g. complex rheology, fluid-structure interaction, and free and moving boundaries. Our formulation rests on full equation coupling and nonlinear solutions achieved through Newton's method with an analytical Jacobian. Furthermore, we advocate the use of LBB-compliant element interpolations for accuracy purposes and make plain in Section 6 the tradeoff of such element types with so-called pressure-stabilized elements developed to circumvent

	Preconditioners					
Element and DOF	ilut(1.0)	ilut(x)	ilut(n) with perturbations	bilu(0)	bilu(x)	bilu(x) with perturbations
Q2Q1 (no RCM) 23,439 DOF	zero pivot	zero pivot	**x** = 1.0 **CN**=9.5e+05 **nnz**=11168721 **A/R**=1.e-5/1.01 70 **its**/315s	**CN** = 1.0e+06 **nnz**=11168721 83 **its**/209s	**n** =1 **CN** = 2.5e06 **nnz**= 15700988 26 **its**/123s	**n**=0 **A/R**=1.e-9/1.0e-9 **CN** = 1.0e06 **nnz**=11168721 83 **its**/209s
Q2Q1 (RCM) 23,439 DOF	**CN** = 7.5e+05 **nnz**=11168721 45 **its**/319s	**x** = 2.0 **CN**=8.6e+05 **nnz**=21544463 2**its**/1331s	**x** = 1.0 **CN**=1.3e+06 **nnz**=11168721 **A/R**=1.e-5/1.0 39 **its**/306s	**CN** = 1.0e+06 **nnz**=11168721 83 **its**/209s	**n** =1 **CN** = 2.5e06 **nnz**= 15700988 26 **its**/123s	**n**=0 **A/R**=1.e-9/1.0e-9 **CN** = 1.0e06 **nnz**=11168721 83 **its**/209s

CN - Condition Number of Preconditioner; **k** - Fill level for bilu;
x - Fill level for ilut; **its** -Average Number of Matrix-solver Iterations.
nnz - Number of Nonzeros Stored (Preconditioner)

Table 6: Convergence results for viscoelastic flow through a four-to-one contraction(∞).

the LBB compatibility condition.

Our approach is unique in that we have pushed it to large, three-dimensional problems in search of an iterative solver/preconditioner strategy that can allow production-level robustness. Heretofore all previous works have made some compromise due to excessive computational work and memory requirements. To this end, and in order to give our approach perspective, we reveiwed in Section 4 every viable alternative within the context of finite element methods, including pressure stabilization methods, problem segregation methods, and penalty methods. We feel this comprehensive review helps justify why our approach today has practical merit and can offer many advantages, but only due to great strides in hardware performance and memory availability over the last few years.

Focusing on GMRES iterative solvers with ILU-based preconditioning, we layed out a strategy in Section 5 that allows successful matrix solution for all but a few cases. Global threshold-based incomplete factorization, viz. ILUT, and block-level incomplete factorization BILU, with threshold perturbations made this success-rate possible, using the estimated preconditioner condition number to help guide the solver parameter choices. In those cases which failed, we have been able to deploy pressure stabilization to a satisfactory level to achieve our desired engineering result. Unfortunately we cannot claim that the successful solution is always a very practical one, as several problems required excessive ILU fill levels that would be prohibitive on a larger scale, cf. viscoelastic flow problems and some fluid-structure interaction problems. Although great strides towards our ultimate goal of achieving iterative solver robustness that rivals the best of direct solvers, but scales favorably with problem size, we admit that our results clearly indicate that much work is still needed.

The conclusions we can draw are as follows:

- We have successfully solved every problem posed to acceptable accuracy using the tactics put forth in Section 5.
- Work still needs to be focused on tactics that allow reduction of the fill level in some problems, especially those involving fluid-structure interactions, as the required fill levels will be impractical for larger problem sizes.
- Threshold perturbations can be used effectively to combat problems exhibiting very large conditions numbers of the preconditioning factors, exceeding approximately 10^{15}. For small to moderate condition numbers, level-of-fill is an effective means for improving convergence for all but free surface problems.
- Reordering a matrix for bandwidth minimization using RCM precipitates condition number growth for multiphysics problems and for problems with discontinuous pressure basis functions. Reordering with RCM is effective at reducing the condition number and improving convergence for continuous Taylor-Hood velocity-pressure elements in incompressible flow problems.
- Variable-block row data structures and BILU preconditioning, although more expensive, is a more robust approach than ILUT.
- Inexactness in the matrix solution process has been proven to improve iterative solver effectiveness and efficiency, but a good initial nonlinear iteration with an accurate matrix solution is essential for problems involving coupled fluid and solid mechanics.
- Pressure-stabilization techniques greatly improve the condition number of the system and allow for accurate solution to low-Re free surface problems. However, for acceptable accuracy on confined flow problems, small stabilization factors are required, small enough to require more fill and perturbations to the preconditioner. Fortunately, our new preconditioner selection approach has allowed us to manage the detrimental effects of PSPG schemes effectively.

Clearly, future work will be focused on block incomplete LU factorization techniques, whereby the block is reduced in size selectively without undermining convergence. Moreover, parallel architectures with the greatly enhance memory availability is perhaps an acceptable way to manage high fill level requirements. Unfortunately this will most likely this will undermine the scalability. Also, we must continue to examine ways to improve performance of this class of preconditioners in three specific cases: discontinuous variable interpolation on an element level, problems involving free surface constraints, and problems involving large bandwidth blocks, as those which involve viscoelasticity.

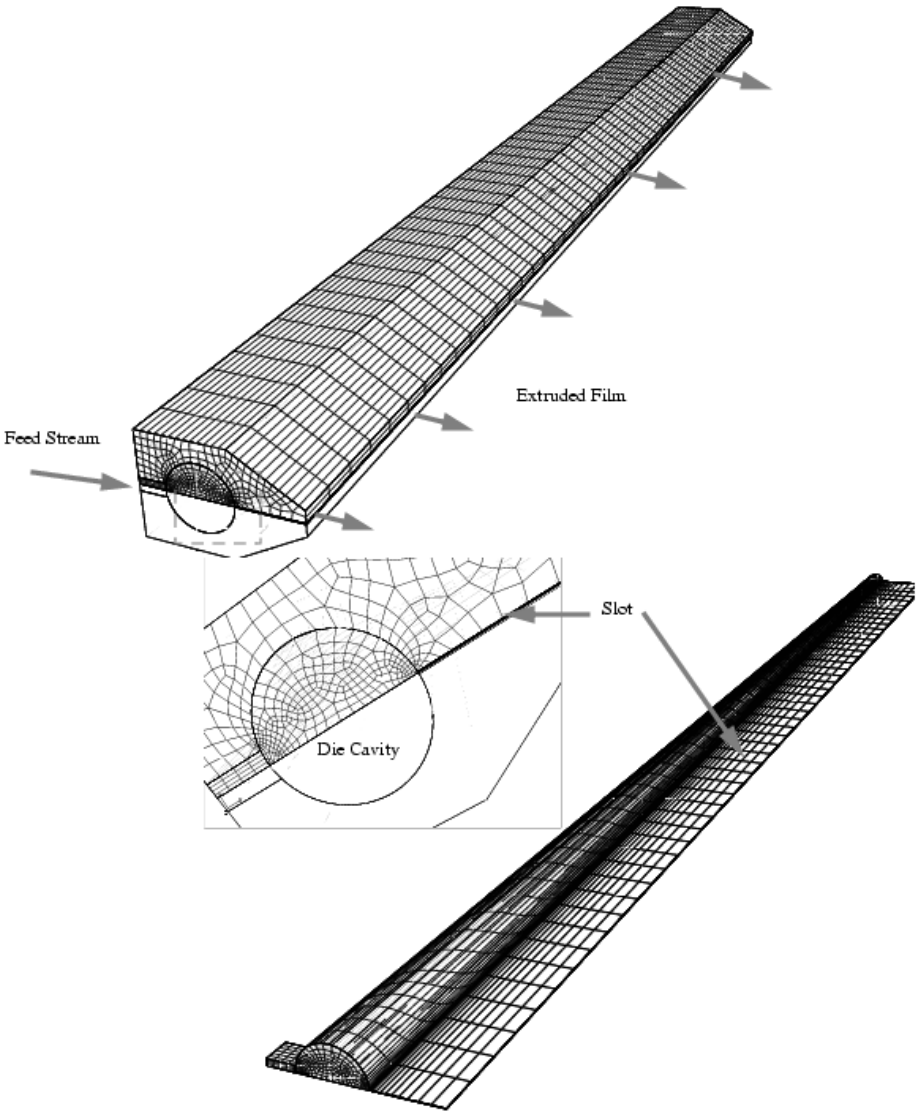

Figure 12: Schematic of die-cavity flexure problem. Total mesh, solid and liquid (top); Flow-only mesh (bottom).

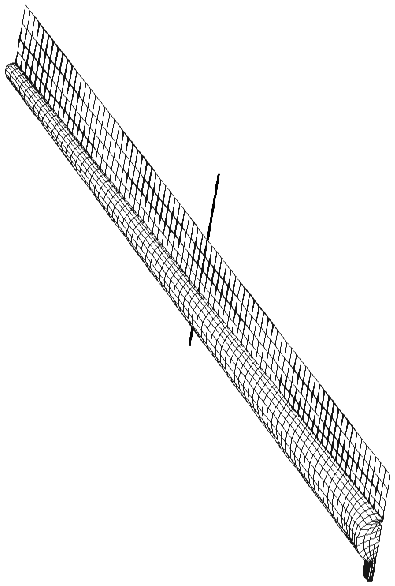

Figure 13: Die-cut grid

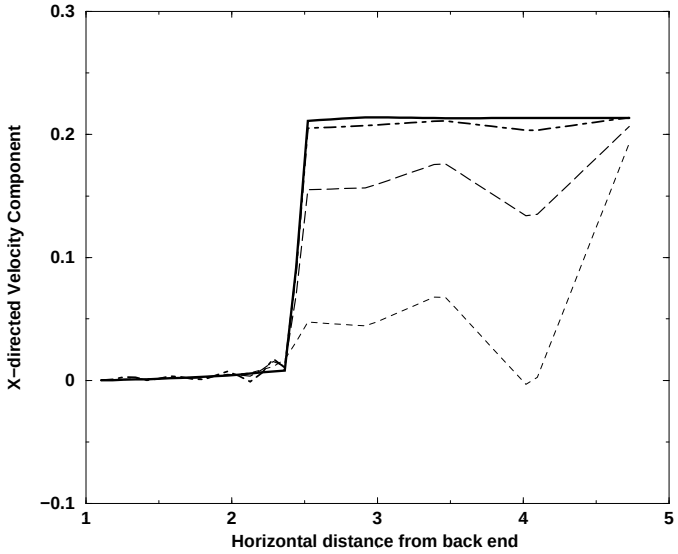

Figure 14: Comparison of x-directed velocity component profile for various levels of the PSPG α parameter, and with a LBB-compliant Q1P0 element

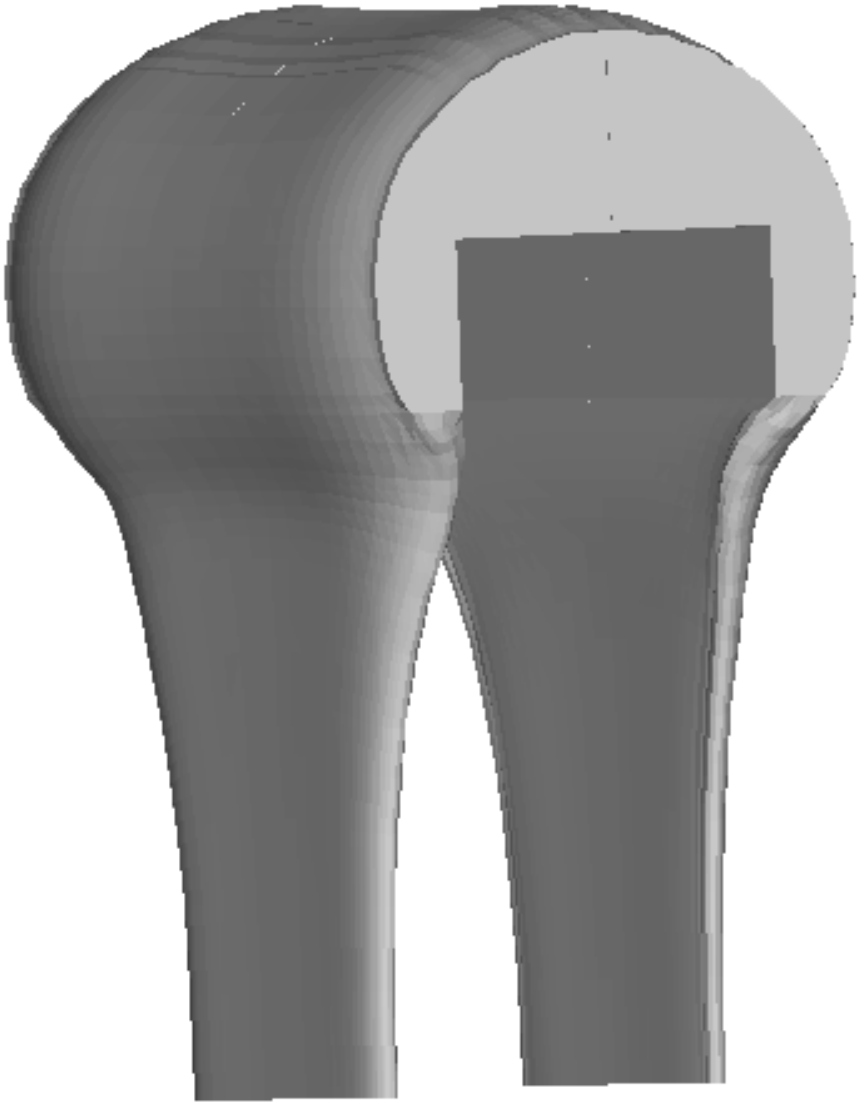

Figure 15: Predicted free surface shape of overflow weir flow.

References

[1] L. W. A. Chapman, Y. Saad, *High-order ilu preconditioners for cfd problems*, Int. J. Numer. Meth. Fluids, 33 (2000), pp. 767–788.

[2] arnold, *whatever, mike*, This that and the other thing, 75 (1994), pp. 119–138.

[3] F. P. T. Baaijens, *Application of low-order discontinuous galerkin method to the analysis of viscoelastic flows*, Journal of Non-Newtonian Fluid Mechanics, 52 (1994), pp. 37–57.

[4] ——, *An iterative solver for DEVSS/DG method with application to smooth and non-smooth flows of the upper convected Maxwell fluid*, Journal of Non-Newtonian Fluid Mechanics, 75 (1998), pp. 119–138.

[5] M. Benzi and M. Tuma, *A sparse approximate inverse preconditioner for nonsymmetric linear systems*, SIAM J. Sci. Comput., 19 (1998), pp. 968–994.

[6] R. B. Bird, R. C. Armstrong, , and O. Hassager, *Dynamics of Polymeric Liquids: Volume 1 Fluid Mechanics*, Wiley, New York, New York, 1997.

[7] F. Brezzi and M. Fortin, *Mixed and Hybrid Finite Element Methods*, Springer-Verlag, New York, 1991.

[8] P. N. Brown and Y. Saad, *Hybrid Krylov methods for nonlinear-systems of equations*, SIAM J. Sci. Stat. Comput., 11 (1990), pp. 450–481.

[9] A. C. Caballero and W. D. Seider, *Large-scale-finite-element numerical simulation of incompressible flows at high levels of convection*, in Presentation at the 1999 AICHE Annual Meeting, Dallas, Texas, November 5 1999, AICHE.

[10] R. A. Cairncross, P. R. Schunk, T. A. Baer, R. R. Rao, and P. A. Sackinger, *A finite element method for free surface flows of incompressible fluids in three dimensions. part i. boundary fitted mesh motion*, International Journal for Numerical Methods in Fluids, 33 (2000), pp. 375–403.

[11] G. F. Carey, K. C. Wang, and W. D. Joubert, *Performance of iterative methods for newtonian and generalized Newtonian flows*, Int. J. Numer. Meth. Fluids, 9 (1989), pp. 127–150.

[12] A. J. Chorin, *A numerical method for solving incompressible viscous flow problems*, J. Comput. Phys, 2 (1967).

[13] E. Chow, *Robust Preconditioning for Sparse Linear Systems*, PhD thesis, University of Minnesota, Minneapolis, MN, 1997.

[14] E. Chow, *A priori sparsity patterns for parallel sparse inverse preconditioners*, SIAM J. Sci. Comput., (2000), pp. 1804–1822.

[15] E. Chow and M. A. Heroux, *An object-oriented framework for block preconditioning*, ACM Transactions on Mathematical Software, 24 (1998), pp. 159–183.

[16] R. Codina and J. Blasco, *A finite element formulation for the Stokes problem allowing equal velocity-pressure interpolation.*, Comput. Methods Appl. Mech. Engrg., 143 (1997), pp. 373–391.

[17] E. H. Cuthill and J. McKee, *Reducing the bandwidth of sparse symmetric matrices*, in Proc. 24^{th} National Conference of the ACM, ACM Press, 1969, pp. 157–172.

[18] J. J. Droux and T. J. R. Hughes, *A boundary integral modification of the Galerkin least squares formulation for the Stokes problem*, Comput. Methods Appl. Mech. Engrg., 113 (1994), pp. 173–182.

[19] F. O. Edis and A. R. Aslan, *Efficient incompressible flow calculations using PQ2Q1 element*, Communications in Numer. Meth. in Engineering, 14 (1998), pp. 161–178.

[20] E. O. Einset and J. F. Jensen, *A finite element solution of three-dimensional mixed convection gas flows in horizontal channels using preconditioned iterative matrix methods*, Int. J. Numer. Meth. Fluids., 14 (1992), pp. 817–841.

[21] M. Engelman, *Fidap: examples manual, revision 6.0*, technical report, Fluid Dynamics International, Evanston, IL, 1991.

[22] M. S. Engelman and I. Hasbani, *Matrix-free solution algorithms in a finite element context*, Report 88-1, Fluid Dynamics International, Evanston, Illinois, 1988.

[23] V. Faber and T. Manteuffel, *Necessary and sufficient conditions for the existence of a conjugate gradient method*, SIAM J. Numer. Anal., 21 (1984), pp. 352–362.

[24] A. Fortin and M. Fortin, *A preconditined generalized minimal residual algorithm for the numerical solution of viscoelastic fluid flows*, J. Non-Newtonian Fluid Mech, 36 (1990), pp. 277–288.

[25] A. Fortin and A. Zine, *An improved GMRES method for solving viscoelastic fluid flow problems*, J. NN Fluid Mechanics, 42 (1992), pp. 1–18.

[26] M. Fortin and A. Fortin, *A new approach for the fem simulations of viscoelastic flow*, Journal of Non-Newtonian Fluid Mechanics, 32 (1989), pp. 295–310.

[27] A. George, *Computer implementation of the finite element method*, Tech. Report STAN-CS-208, Stanford University, 1971.

[28] G. H. Golub and C. F. V. Loan, *Matrix Computations*, John Hopkins University Press, Baltimore, Md., 3rd ed., 1996.

[29] P. M. Gresho, *Some current CFD issues relevant to the incompressible Navier Stokes equations*, Report UCRL-JC-104073 Rev.1, Lawrence Livermore National Laboratory, 1990.

[30] R. Guenette and M. Fortin, *A new mixed finite element method for computing viscoelastic flow*, Journal of Non-Newtonian Fluid Mechanics, 60 (1995), pp. 27–52.

[31] V. Haroutunian, M. S. Engelman, and I. Hasbani, *Segregated finite element algorithms for the numerical solution of large-scale incompressible flow problems*, Int. J. Numer. Meth. Fluids, 17 (1993), pp. 323–348.

[32] Y. Hasbani and M. Engelman, *Out-of-core solution of linear equations with non-symmetric coefficient matrix*, Comput. Fluids., 7 (1979), pp. 13–31.

[33] M. Heroux, *IFPACK 2.0: An object-oriented parallel scaled incomplete factorization package*, Tech. Report SAND2002-XXXX, Sandia National Laboratories, Albq, NM, 2002.

[34] P. Hood, *Frontal solution program for unsymmetric matrices*, Int. J. Numer. Meth. Eng., 10 (1976), pp. 379–399.

[35] D. Howard, W. M. Connolley, and J. S. Rollet, *Unsymmetric conjugate gradient methods and sparse direct methods in finite element flow simulations*, Int. J. Numer. Meth. Fluids., 10 (1990), pp. 925–945.

[36] A. Huerta and W. K. Liu, *Viscous flows with large free surface motion*, Comput. Meth. Appl. Mech. Eng., 69 (1988), pp. 277–324.

[37] T. J. R. Hughes, L. P. Franca, and M. Balestra, *A new finite element formulation for computational fluid dynamics: V circumventing the Babuska-Brezzi condition: A stable Petrov-Galerkin formulation of the Stokes problem accommodating equal-order interpolations*, Comput. Meth. Appl. Mech. Eng., 59 (1986), pp. 85–99.

[38] T. R. J. Hughes and L. P. Franca, *A new finite element formulation for computational fluid dynamics: VII the stokes problem with various well-posed boundary conditions: Symmetric formulations that converge for all velocity/pressure spaces*, Comput. Methods Appl. Mech. Engrg, 65 (1987), pp. 85–96.

[39] F. Inc, *Fidap 8 manuals*, Fluent, Inc. Lebanon, NH, 1-8 (1998).

[40] H. S. Khesghi and M. Luskin, *On the variable sign penalty method for the computation of solutions to the Navier-Stokes equations*, in Contemporary Mathematics, J. A. Smoller, ed., vol. 17, Amer. Math. Soc., Providence, Rhode Island, 1983.

[41] D. A. Knoll, D. B. Kothe, and B. Lally, *A new nonlinear solution method for phase-change problems*, Numerical Heat Transfer Part B-Fundamentals, 35 (1999), pp. 439–459.

[42] L. Y. Kolotilina and A. Y. Yeremin, *Factorized sparse approximate inverse preconditionings. I. Theory*, SIAM Journal on Matrix Analysis and Applications, 14 (1993), pp. 45–58.

[43] T. A. Manteuffel, *An incomplete factorization technique for positive definite linear systems*, Math. Comp., 34 (1980), pp. 473–497.

[44] J. E. Marsden and J. R. Hughes., *Mathematical Foundations of Elasticity*, Prentice-Hall, Englewood Cliffs, NJ., 1983.

[45] S. V. Patankar and D. B. Spalding, *A calculation procedure for heat, mass, and momentum transfer in three-dimensional flows*, Int. J. of Heat and Mass Transfer, 15 (1976), pp. 1787–1806.

[46] D. Rajagolan, R. C. Armstrong, and R. A. Brown, *Finite element methods for calculation of viscoelastic fluids with a newtonian viscosity*, Journal of Non-Newtonian Fluid Mechanics, 36 (1990), pp. 159–192.

[47] J. N. Reddy and D. K. Gartling, *The Finite Element Method in Heat Transfer and Fluid Dynamics*, CRC Press, Boca Raton, Fl., 1994.

[48] Y. Saad, *Preconditioning techniques for nonsymmetric and indefinite linear-systems*, J. Comp. Appl. Math., 24 (1988), pp. 89–108.

[49] ——, *ILUT: A dual threshold incomplete lu preconditioner*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.

[50] Y. Saad, *Preconditioned krylov subspace methods for CFD applications*, in Proceedings of the International Workshop on Solution Techniques for Large-scale CFD problems, W. G. Habashi, ed., 1994, pp. 179–195.

[51] Y. Saad, *Iterative methods for sparse linear systems*, PWS Publishing, New York, 1996.

[52] Y. Saad and M. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.

[53] P. A. Sackinger, P. R. Schunk, and R. R. Rao, *A Newton-Raphson pseudo-solid domain mapping technique for free and moving boundary problems: a finite element implementation*, J. Comp. Physics, 125 (1996), pp. 83–103.

[54] A. G. Salinger, Q. Xiao, Y. Zhou, and J. J. Derby, *Massively parallel finite element computations of three-dimensional, time-dependent, incompressible flow in materials processing systems*, Comput. Methods Appl. Mech. Engrg, 149 (1994), pp. 139–156.

[55] J. N. Shadid, R. S. Tuminaro, and H. F. Walker, *An inexact Newton method for fully-coupled solution of the Navier Stokes equations with heat and mass transport*, Technical Report SAND97-0132, Sandia National Laboratories, Albuquerque, NM, 1997.

[56] D. Silvester, H. Elman, D. Kay, and A. Wathen, *Efficient preconditioning of the linearized Navier-Stokes equations*, Tech. Report 352, Manchester, England, 1999.

[57] J. Strigberger, G. Baruzzi, W. Habashi, and M. Fortin, *Some special purpose preconditioners for conjugate gradient-like methods applied to CFD*, Int. J. Numer. Methods in Fluids, 16 (1993), pp. 581–596.

[58] T. E. Tezduyar, R. Shih, S. Mittal, and S. E. Ray, *Incompressible flow computations with stabilized bilinear and linear equal order interpolation velocity-pressure elements*, Research Report UMSI 90/165, University of Minnesota Supercomputer Institute, Minneapolis, Mn., 1990.

[59] C. C. Tsai and T. J. Liu., *Comparison of three solvers for viscoelastic fluid flow problems*, J. Non-Newtonian Fluid Mech., 60 (1995), pp. 155–177.

[60] R. Tuminaro and C. Tong, *Parallel smoothed aggregation multigrid: Aggregation strategies on massively parallel machines*, in SuperComputing 2000 Proceedings, J. Donnelley, ed., 2000.

[61] R. S. Tuminaro, M. A. H. S. A. Hutchinson, and J. N. Shadid, *Official Aztec User's Guide, Version 2.1*, Sandia National Laboratories, Albuquerque, NM 87185, 1999.

[62] H. van der Vorst, *Iterative solution methods for certain sparse linear systems with a non-symmetric matrix arising from PDE-problems*, Journal of Computational Physics, 44 (1981), pp. 1–19.

[63] ——, *Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 631–644.

[64] G. A. Watson, ed., *Constrained variational principles and penalty function methods in finite element analysis*, Lecture Notes in Mathematics: Conference on the Numerical Solution of Differential Equations, Berlin, 1974, Springer-Verlag.

[65] J. Zhang., *Preconditioned krylov subspace methods for solving nonsymmetric matrices from cfd applications*, Compt. Methods Appl. Mech. Engrg, 189 (2000), pp. 825–840.

Distribution:

MS 0612	4912	Review and Approval Desk for DOE/OSTI
MS 0899	4916	Technical Library (2)
MS 0719	6131	Eric R. Lindgren
MS 9405	8700	T. M. Dyer
MS 9042	8728	M. W. Perra
MS 9042	8728	Greg H. Evans
MS 9042	8728	William G. Houf
MS 9042	8728	Mike P. Kanouff
MS 9051	8728	Ron Larson
MS 9042	8728	Chris Moen
MS 0827	8920	Alan B. Williams
MS 9018	8940-2	Central Technical Files
MS 0828	9100	Thomas C. Bickel
MS 0826	9100	David K. Gartling
MS 0834	9100	Melvin A. Baer
MS 0834	9100	T. Y. Chu
MS 0836	9100	Carl Peterson
MS 0828	9102	Robert K. Thomas
MS 0841	9102	Joseph Fernandez
MS 0825	9111	Steven N. Kempka
MS 0825	9111	Steve Bova
MS 0825	9111	Shawn Burns
MS 0825	9111	Robert J. Cochran
MS 0825	9111	Micheal W. Glass
MS 0825	9111	Basil Hassan
MS 0825	9111	Randy L. Lober
MS 0825	9111	Philip A. Sackinger (3)
MS 0825	9111	Samuel R. Subia (3)
MS 0834	9112	Arthur C. Ratzel
MS 0826	9113	Wahid Hermina

(cont'd next page)

Distribution (cont'd):

MS 0826	9113	David R. Noble
MS 0826	9113	John R. Torczynski
MS 0826	9113	C. Channy Wong
MS 0834	9114	Justine Johannes
MS 0834	9114	Thomas A. Baer
MS 0834	9114	Ken Chen
MS 0834	9114	Richard C. Givler
MS 0834	9114	Matt Hopkins
MS 0834	9114	Polly L. Hopkins
MS 0834	9114	Duane Labreche (10)
MS 0834	9114	Mario J. Martinez
MS 0601	9114	Harry K. Moffatt
MS 0834	9114	Rekha Rao
MS 0834	9114	R. Allen Roach
MS 0834	9114	P. Randall Schunk
MS 0834	9114	Amy Sun
MS 0825	9115	Walt H. Rutledge
MS 0836	9116	Eugene Hertel
MS 0836	9117	Richard Griffith
MS 0836	9117	Steve Gianoulakis
MS 0836	9117	Roy Hogan
MS 0835	9121	James S. Peery
MS 0847	9123	Hal Morgan
MS 0847	9124	Dave Martinez
MS 0847	9124	Todd Simmermacher
MS 0827	9131	John D. Zepper
MS 0827	9131	Carter Edwards
MS 0827	9131	Thomas Otahal
MS 0827	9131	Larry Schoof
MS 0827	9131	Greg Sjaardema
MS 0827	9131	James Stewart
MS 0828	9132	Jaime Moya

(cont'd next page)

Distribution (cont'd):

MS 0828	9133	Martin Pilch
MS 0321	9200	William Camp
MS 1111	9221	Sudip S. Dosanjh
MS 1111	9221	Gary L. Hennigan
MS 1111	9221	Andy Salinger
MS 1111	9221	Scott A. Hutchinson
MS 1111	9221	John N. Shadid
MS 1110	9222	David Womble
MS 1110	9222	Mike Heroux
MS 1110	9222	Richard Lehoucq
MS 9214	9222	Ray Tuminaro
MS 1111	9226	Rob Leland
MS 1111	9226	Bruce A. Hendrickson
MS 1110	9223	Neil D. Pundit
MS 0819	9231	Ed Boucheron
MS 0819	9231	J. Randy Weatherby